Identificación de los datos del proyecto:

- **Tema:** APLICACIONES DE RECONOCIMIENTO DE VOZ EN EL APRENDIZAJE DE LAS LENGUAS
- **Título:** ACTIVIDADES ONLINE DE VOCABULARIO Y PRONUNCIACIÓN PARA EL NIVEL DE LA LENGUA INGLESA A2
- **Autor:** MIGUEL ÁNGEL DOMÍNGUEZ MORLA
- **Titulación:** TELEMÁTICA
- **Tutor:** IRINA ARGÜELLES ÁLVAREZ
- **Departamento:** LINGÜÍSTICA APLICADA
- **Director:**
- **Tribunal:**
- **Presidente:** ANA GÓMEZ OLIVA
- **Vocal:** IRINA ARGÜELLES ÁLVAREZ
- **Vocal Secretario:** INMACULADA ÁLVAREZ DE MON REGO
- **Fecha de lectura:** 30 de Septiembre de 2013

RESUMEN DEL PROYECTO:

El objetivo del presente proyecto es proporcionar una actividad de la pronunciación y repaso de vocabulario en lengua inglesa para la plataforma Moodle alojada en la página web de *Integrated Language Learning Lab* (ILLLab). La página web ILLLab tiene el objetivo de que los alumnos de la EUIT de Telecomunicación de la UPM con un nivel de inglés A2 según el Marco Común Europeo de Referencia para las Lenguas (MCERL), puedan trabajar de manera autónoma para avanzar hacia el nivel B2 en inglés. La UPM exige estos conocimientos de nivel de inglés para cursar la asignatura *English for Professional and Academic Communication* (EPAC) de carácter obligatorio e impartida en el séptimo semestre del Grado en Ingeniería de Telecomunicaciones.

Asimismo, se persigue abordar el problema de las escasas actividades de expresión oral de las plataformas de autoaprendizaje se dedican a la formación en idiomas y, más concretamente, al inglés. Con ese fin, se proporciona una herramienta basada en sistemas de reconocimiento de voz para que el usuario practique la pronunciación de las palabras inglesas.

En el primer capítulo del trabajo se introduce la aplicación *Traffic Lights*, explicando sus orígenes y en qué consiste.

En el segundo capítulo se abordan aspectos teóricos relacionados con el reconocimiento de voz y se comenta sus funciones principales y las aplicaciones actuales para las que se usa.

El tercer capítulo ofrece una explicación detallada de los diferentes lenguajes utilizados para la realización del proyecto, así como de su código desarrollado.

En el cuarto capítulo se plantea un manual de usuario de la aplicación, exponiendo al usuario cómo funciona la aplicación y un ejemplo de uso. Además, se añade varias secciones para el administrador de la aplicación, en las que se especifica cómo agregar nuevas palabras en la base de datos y hacer cambios en el tiempo estimado que el usuario tiene para acabar una partida del juego.

ABSTRACT:

The objective of the present project is to provide an activity of pronunciation and vocabulary review in English language within the platform Moodle hosted at the *Integrated Language Learning Lab* (ILLLab) website. The ILLLab website has the aim to provide students at the EUIT of Telecommunication in the UPM with activities to develop their A2 level according to the Common European Framework of Reference for Languages (CEFR). In the platform, students can work independently to advance towards a B2 level in English. The UPM requires this level of English proficiency for enrolling in the compulsory subject *English for Professional and Academic Communication* (EPAC) taught in the seventh semester of the Degree in Telecommunications Engineering.

Likewise, this project tries to provide alternatives to solve the problem of scarce speaking activities included in the learning platforms that offer language courses, and specifically, English language courses. For this purpose, it provides a tool based on speech recognition systems so that the user can practice the pronunciation of English words.

The first chapter of the project introduces the application *Traffic Lights*, explaining its origins and what it is.

The second chapter deals with theoretical aspects related with speech recognition and comments their main features and current applications for which it is generally used.

The third chapter provides a detailed explanation of the different programming languages used for the implementation of the project and reviews its code development.

The fourth chapter presents an application user manual, exposing to the user how the application works and an example of use. Also, several sections are added addressed to the application administrator, which specify how to add new words to the database and how to make changes in the original stings as could be the estimated time that the user has to finish the game.

Agradecimientos

Este proyecto supone la finalización de una etapa importante en mi vida. A partir de ahora comienza un camino lleno de esfuerzo a la vez que, espero, de satisfacción.

Agradezco a mis padres y mis hermanos, que han hecho posible la realización de esta carrera universitaria. En especial, doy gracias mi hermano Javier, que me ha apoyado y animado en todo momento a realizar mis sueños.

También les doy gracias a mis amigos me acompañan desde toda mi vida. Ellos me han ayudado a realizar este proyecto, en pequeña o gran medida han aportado su granito de arena y eso refleja un gran gesto de su parte.

Asimismo, quiero expresar mi gratitud a mis amigos y compañeros de la universidad. Con ellos he pasado buenos recuerdos que nunca olvidare y espero seguir creando más momentos con ellos.

Finalmente, gracias a mi tutora Irina, que ha sido muy agradable realizar este proyecto con ella, no me ha presionado en ningún momento y siempre me ha ofrecido ayuda para terminar este proyecto.

De todo corazón, muchas gracias.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

Contenido **Página**

RESUMEN DEL PROYECTO:

El objetivo del presente proyecto es proporcionar una actividad de la pronunciación y repaso de vocabulario en lengua inglesa para la plataforma Moodle alojada en la página web de *Integrated Language Learning Lab* (ILLLab). La página web ILLLab tiene el objetivo de que los alumnos de la EUIT de Telecomunicación de la UPM con un nivel de inglés A2 según el Marco Común Europeo de Referencia para las Lenguas (MCERL), puedan trabajar de manera autónoma para avanzar hacia el nivel B2 en inglés. La UPM exige estos conocimientos de nivel de inglés para cursar la asignatura *English for Professional and Academic Communication* (EPAC) de carácter obligatorio e impartida en el séptimo semestre del Grado en Ingeniería de Telecomunicaciones.

Asimismo, se persigue abordar el problema de las escasas actividades de expresión oral de las plataformas de autoaprendizaje se dedican a la formación en idiomas y, más concretamente, al inglés. Con ese fin, se proporciona una herramienta basada en sistemas de reconocimiento de voz para que el usuario practique la pronunciación de las palabras inglesas.

En el primer capítulo del trabajo se introduce la aplicación *Traffic Lights*, explicando sus orígenes y en qué consiste.

En el segundo capítulo se abordan aspectos teóricos relacionados con el reconocimiento de voz y se comenta sus funciones principales y las aplicaciones actuales para las que se usa.

El tercer capítulo ofrece una explicación detallada de los diferentes lenguajes utilizados para la realización del proyecto, así como de su código desarrollado.

En el cuarto capítulo se plantea un manual de usuario de la aplicación, exponiendo al usuario cómo funciona la aplicación y un ejemplo de uso. Además, se añade varias secciones para el administrador de la aplicación, en las que se especifica cómo agregar nuevas palabras en la base de datos y hacer cambios en el tiempo estimado que el usuario tiene para acabar una partida del juego.

ABSTRACT:

The objective of the present project is to provide an activity of pronunciation and vocabulary review in English language within the platform Moodle hosted at the *Integrated Language Learning Lab* (ILLLab) website. The ILLLab website has the aim to provide students at the EUIT of Telecommunication in the UPM with activities to develop their A2 level according to the Common European Framework of Reference for Languages (CEFR). In the platform, students can work independently to advance towards a B2 level in English. The UPM requires this level of English proficiency for enrolling in the compulsory subject *English for Professional and Academic Communication* (EPAC) taught in the seventh semester of the Degree in Telecommunications Engineering.

Likewise, this project tries to provide alternatives to solve the problem of scarce speaking activities included in the learning platforms that offer language courses, and specifically, English language courses. For this purpose, it provides a tool based on speech recognition systems so that the user can practice the pronunciation of English words.

The first chapter of the project introduces the application *Traffic Lights*, explaining its origins and what it is.

The second chapter deals with theoretical aspects related with speech recognition and comments their main features and current applications for which it is generally used.

The third chapter provides a detailed explanation of the different programming languages used for the implementation of the project and reviews its code development.

The fourth chapter presents an application user manual, exposing to the user how the application works and an example of use. Also, several sections are added addressed to the application administrator, which specify how to add new words to the database and how to make changes in the original stings as could be the estimated time that the user has to finish the game.

INTRODUCCIÓN

Hoy en día, el aprendizaje de la lengua inglesa es muy importante en el ámbito mundial. La mayoría de países la utilizan para comunicarse entre ellos, no sólo profesionalmente sino también socialmente. Se trata del tercer idioma más hablado en el mundo y, por su facilidad de aprender comparado con los demás idiomas debido a que muchos derivan del latín, se ha convertido en el lenguaje por excelencia, según estadísticas realizadas por "The Ethnologue".

Por ello, la Universidad Politécnica de Madrid quiere que sus alumnos tengan un conocimiento óptimo a la hora de entrar en sus aulas, para que puedan comprender y desempeñar las actividades de sus asignaturas sin dificultad, de modo que cuando terminen sus estudios puedan optar por un empleo y futuro mejor.

Así que la Universidad Politécnica de Madrid se encarga de que los alumnos puedan acceder a la universidad con un conocimiento de inglés mínimo exigido y si no lo alcanzan, el Departamento de Lingüística Aplicada a la Ciencia y a la Tecnología de la Escuela Universitaria de Ingeniería Técnica de Telecomunicaciones les ofrece una serie de herramientas y actividades para alcanzar este nivel exigido.

Motivación

La asignatura de séptimo semestre del Grado en Ingeniería de Telecomunicaciones, English for Professional and Academic Communication (EPAC) exige como requisitos previos, además de otros requerimientos, el nivel B2 de inglés según el Marco Común Europeo de Referencia para las Lenguas (MCERL) y es, asimismo, el nivel mínimo que muchas universidades europeas establecen como requisito para los estudiantes que participan en programas de intercambio, como Erasmus.

Por ello es necesaria la acreditación del nivel B2 que podrá realizarse en organismos oficiales externos a la UPM, o mediante pruebas que al efecto diseñe la UPM a través de su Departamento de Lingüística Aplicada a la Ciencia y Técnología. En septiembre de 2009, 255 estudiantes de la EUITT realizaron una prueba teórica llevada a cabo por el departamento de inglés de la EUITT, que demostró que el 65% de los alumnos presentados no habían llegado a un nivel intermedio de inglés B2.

Por esta razón, el departamento de inglés creó una plataforma e-learning para ayudar a los alumnos a alcanzar un nivel B1 de inglés para la universidad llamada ILLLab (Integrated Language Learning Lab) que comprenden desde un moodle, que es una aplicación web de tipo ambiente educativo virtual, con un sistema de gestión de cursos de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea, hasta un blog con actividades interesantes como conferencias y visionado de películas en inglés.

De esta forma, los alumnos pueden **practicar su autoaprendizaje** para alcanzar el nivel deseado B2 exigido para cursas la asignatura de grado EPAC y debido a la escasez de actividades orales individuales de libre distribución ofrecidas en internet o en la universidad, surge este proyecto.

Objetivos

El objetivo del presente proyecto es ayudar a mejorar a los alumnos de nivel A2 con la pronunciación en lengua inglesa, además de que los estudiantes repasen o incorporen nuevos términos en su vocabulario.

Para cubrir estas necesidades, al menos en parte, el Departamento de Lingüística Aplicada a la Ciencia y a la Tecnología propone este proyecto, una actividad de vocabulario en forma de juego, con el fin de que los alumnos practiquen de forma amena su nivel de conocimientos, además de ejercitar su pronunciación con un sistema de reconocimiento de voz sencillo.

La aplicación se implementará en ILLLab (Integrated Learning Language Lab), una plataforma web desarrollada por Departamento de Lingüística Aplicada a la Ciencia y a la Tecnología de la UPM, que tiene como objetivo principal fomentar el aprendizaje de la lengua inglesa a través de diferentes actividades didácticas y recursos lingüísticos disponibles on-line.

1. CAPÍTULO 1: TRAFFIC LIGHTS

Originalmente se pensó en un juego parecido al concurso televisivo *"Pasapalabra"* aunque con algunas modificaciones y sin las pruebas previas que proponen el concurso original y el juego de mesa. Asimismo, se deseaba añadir una aplicación en inglés de autoaprendizaje en la plataforma ILLLab, para que los estudiantes practicaran su vocabulario y su pronunciación.

De ahí, surgió *"Traffic Lights"*, una aplicación basada en pasapalabra, que es un tipo de juego de alfabeto. En "*Traffic Lights"* hay que averiguar las palabras a partir de su definición y de su letra inicial, recorriendo letra por letra el abecedario, desde la A hasta la Z. Asimismo, se generaría una lista aleatoria de palabras para cada vez que el usuario iniciará una partida, para dar versatilidad al juego.

Cada letra tiene su propio semáforo, el cual se irá encendiendo según el jugador conteste correctamente a la palabra o no, o si éste ha decidido pasar a la siguiente palabra. Por ejemplo, el semáforo se pondrá en verde si el jugador ha contestado correctamente a la palabra, o en rojo si la palabra ha sido incorrecta. En caso de pasar a la siguiente palabra, el semáforo se pondrá en ámbar, indicando que esa palabra podrá ser contestada a la siguiente ronda.

Además, se añade un temporizador de cuenta atrás y se agregan unas reglas, en las que cada vez que el usuario contesta una palabra, afecta al temporizador, suponiendo un desafío mayor para el usuario. Por ejemplo, si se contesta bien a una palabra se añadirán 5 segundos al temporizador del usuario, en caso contrario, se le restarán 5 segundos.

Por último, la aplicación posee un complemento de reconocimiento de voz, para que el jugador pronuncie la palabra que cree correcta para la letra en cuestión. De este modo, la aplicación la traduce a texto para escribirla en una caja de texto que posee la aplicación.

Para desarrollar "*Traffic Lights"*, se necesitaba una base de datos de palabras junto con sus definiciones, por lo que se buscó en internet bases de datos con estas características. Se encontraron bases de datos de palabras con definiciones. Se halló una conocida base de datos realizada por la universidad de Princeton llamada WordNet, que era la que más se adaptaba a nuestra aplicación.

WordNet es una gran base de datos léxica en inglés. En esta base de datos; sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos llamados *synsets*. Estos *synsets* están vinculados entre sí mediante relaciones conceptuales, semánticas y léxicas. La red resultante de palabras puede ser utilizada desde el navegador web. Además, WordNet es de acceso libre y público, por lo que puede descargarse desde su página web http://wordnet.princeton.edu/wordnet/. En un principio, nos dispusimos a utilizarla para la aplicación.

Entonces surgieron varios problemas para añadir esta base de datos a nuestra aplicación.

En primer lugar, era demasiado extensa, lo que significaba que tardaría mucho en generar la lista aleatoria de palabras de la base de datos que utilizaríamos para cada partida, haciendo que el usuario tuviera que esperar hasta que la lista estuviera terminada para comenzar a jugar.

A parte de este problema, las definiciones no estaban pensadas para alumnos con un conocimiento de nivel de inglés A2, por lo que al leerlas e intentar comprenderlas, supondría un esfuerzo mucho mayor y acabarían desistiendo.

Igualmente, la base de datos contenía muchos más datos de los que necesitábamos. Por lo que ocuparíamos nuestra base de datos con datos inútiles, puesto que no los utilizaríamos para nuestra aplicación.

Por estas razones, decidimos hacer nuestra propia base de datos. La base de datos contiene un total de 130 palabras, cinco palabras de cada letra del alfabeto inglés. Esta base de datos es una propuesta inicial y se puede ampliar añadiendo nuevas palabras y definiciones que se adapten al nivel que se desea impartir en la aplicación.

Para saber qué tipos de palabras incluir en la base de datos, se hizo una estimación de las *categorías de palabras* que se utilizan en tres textos generales, sacados del libro *HeadWay Student's Book* de la editorial *Oxford University Press*.

Al igual que en WordNet, se decidió incluir las cuatro categorías importantes: nombres, verbos, adjetivos y adverbios. Se hizo estimación de la estadística del porcentaje de las palabras de cada categoría que aparecen en los textos con los siguientes resultados:

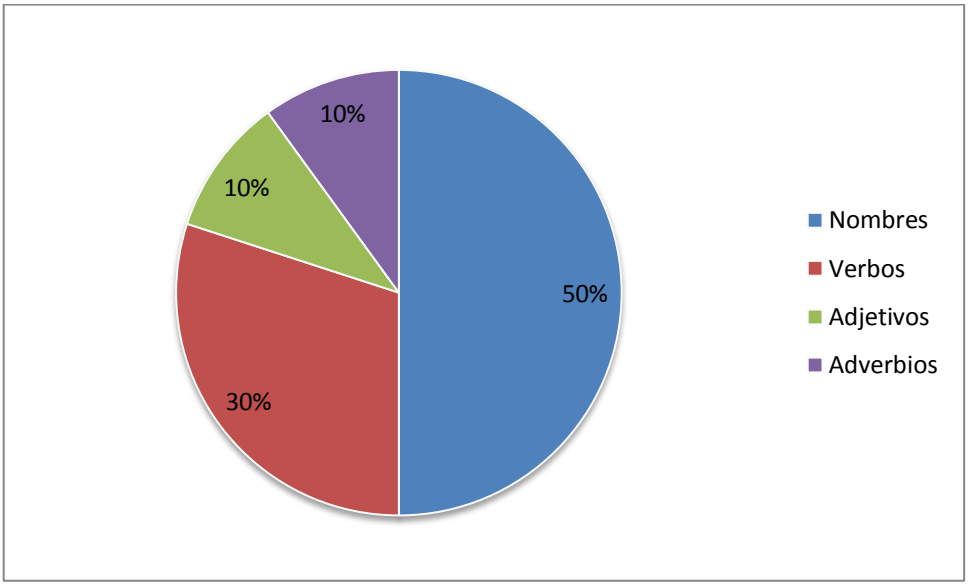


Figura 2.1: Estimación de palabras utilizadas en textos generales (en %)

De esta forma, introducimos el siguiente número de palabras según cada categoría en la base de datos mostrado en la figura 3.2:

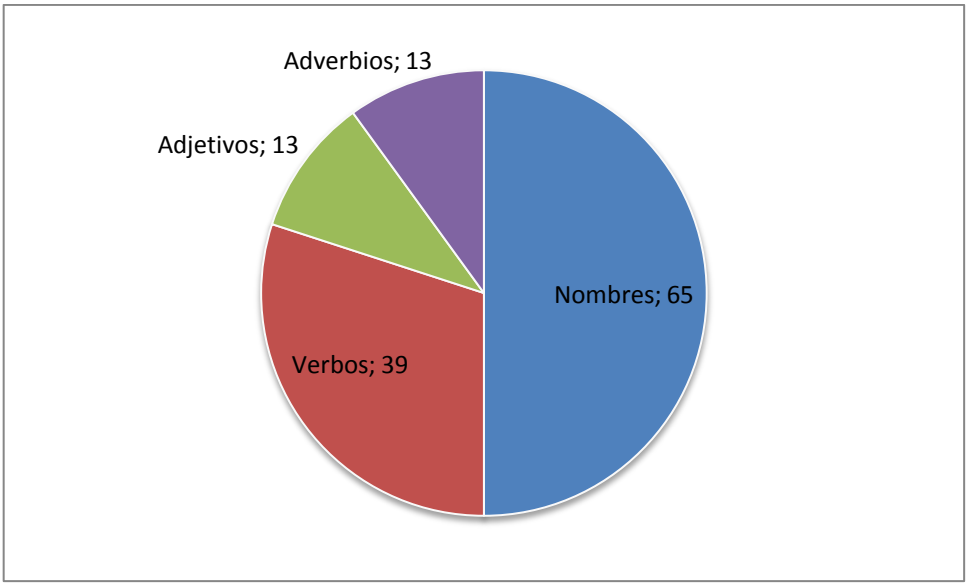


Figura 2.2: Estimación de palabras utilizadas en textos generales

Para realizar las definiciones legibles y claras, se dispuso de diccionarios online intentándose ajustar al criterio del nivel de inglés A2. Los diccionarios online utilizados son los siguientes:

http://www.wordreference.com/

http://www.thefreedictionary.com/

http://dictionary.cambridge.org/

2. CAPÍTULO 2: RECONOCIMIENTO DE VOZ

A pesar del aparente éxito de estas tecnologías, muy pocas personas utilizan el sistema del reconocimiento del habla en sus ordenadores. Parece ser que muchos de los usuarios utilizan el ratón y el teclado para guardar o redactar documentos, porque les resulta más cómodo y rápido a pesar del hecho de que todos podemos hablar a más velocidad de la que tecleamos. Sin embargo, mediante el uso de ambos, el teclado y el reconocimiento del habla, nuestro trabajo sería mucho más efectivo.

Este sistema donde está siendo más utilizado es en aplicaciones telefónicas: agencias de viajes, atención al cliente, información etc. También se han ido instaurando en aplicaciones web como en los buscadores web o en dictados de voz a texto. La mejoría de estos sistemas de reconocimiento del habla han ido aumentando y su eficacia cada vez es mayor.

2.1. Definición de reconocimiento de voz

El proceso de reconocimiento de voz o reconocimiento automático del habla (RAH) dota a las máquinas de la capacidad de recibir mensajes orales. Tomando con entrada la señal acústica recogida por un micrófono, el proceso de reconocimiento automático del habla tiene como objetivo final descodificar el mensaje contenido en la onda acústica para realizar las acciones pertinentes.

Para lograr este fin, un **sistema de RAH** necesitaría conjugar una gran cantidad de conocimientos acerca del sistema auditivo humano, sobre la estructura del lenguaje, la representación del significado de los mensajes y sobre todo el autoaprendizaje de la experiencia diaria.

Actualmente estamos lejos de lograr un sistema completo que pueda comprender cualquier mensaje oral en cualquier contexto tal y como lo podría hacer un ser humano. Sin embargo, la tecnología actual sí que permite realizar sistemas de RAH que pueden trabajar, con un error aceptable, en entornos semánticos restringidos.

Básicamente, el reconocimiento del habla es un proceso de clasificación de patrones, cuyo objetivo es clasificar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos diccionarios de modelos acústicos y de lenguaje.

Este proceso de clasificación supone, en primer lugar, que la señal de voz puede ser analizada en segmentos de corta duración y representar cada uno de los segmentos mediante su contenido frecuencial, de forma análoga al funcionamiento del oído.

En segundo lugar, mediante un proceso de clasificación podemos asignar a cada segmento o conjuntos consecutivos de segmentos una unidad con significado lingüístico.

Finalmente, en tercer lugar, mediante un procesador lingüístico podemos dar significado a las secuencias de unidades. Este último paso del sistema supone incorporar al sistema de RAH conocimiento acerca de la estructura sintáctica, semántica y pragmática del lenguaje.

Sin embargo, los sistemas actuales de RAH solo incorporan estas fuentes de conocimiento sobre tareas muy restringidas y controladas, estando la mayoría de ellos en experimentación en condiciones de laboratorio.

Un aspecto crucial en el diseño de un sistema de RAH es la elección del tipo de **aprendizaje** que se utilice para construir las diversas fuentes de conocimiento que serán las bases del sistema de RAH. Básicamente, existen dos tipos:

- **Aprendizaje deductivo**: Las técnicas de aprendizaje deductivo se basan en la transferencia de los conocimientos que un ser humano experto posee a un sistema informático.
- **Aprendizaje inductivo**: Las técnicas de aprendizaje inductivo se basan en que el sistema pueda, automáticamente, conseguir los conocimientos necesarios a partir de ejemplos reales sobre la tarea que se desea modelizar.

En la práctica, no existen metodologías que estén basadas únicamente en el aprendizaje inductivo, de hecho, se asume un compromiso deductivo-inductivo en el que los aspectos generales se suministran deductivamente y la caracterización de la variabilidad inductivamente.

Las fuentes de información acústica, fonética, fonológica y posiblemente léxica, con los correspondientes procedimientos interpretativos, dan lugar a un módulo conocido como **decodificador acústico-fonético**. La entrada al decodificador acústico-fonético es la *señal vocal* convenientemente representada; para ello, es necesario que ésta sufra un pre-proceso de parametrización. En esta etapa previa es necesario asumir algún modelo físico, contándose con modelos auditivos y modelos articulatorios.

Las fuentes de conocimiento sintáctico, semántico y pragmático dan lugar al **modelo del lenguaje** del sistema. Cuando la representación de la sintaxis y de la semántica tiende a integrarse, se desarrollan sistemas de RAH de gramática restringida para tareas concretas.

El **reconocimiento de la gramática restringida** trabaja reduciendo las típicas frases reconocidas a un tamaño más pequeño que la gramática formal. Este tipo de reconocimiento trabaja mejor cuando el hablante proporciona respuestas breves a cuestiones o preguntas específicas: las preguntas de "si" o "no", al elegir una opción del menú, un artículo de una lista determinada, etc. La gramática específica las palabras y frases más típicas que una persona diría como respuesta rápida y después asocia esas palabras o frases a un concepto semántico. Por ejemplo, un "si" puede entenderse cuando se oye un "sip", "vale", "yes" o "okey", y un "no" con un "nop", "nada" o "en absoluto".

Si el hablante dice algo que gramaticalmente no tiene sentido, el reconocimiento fallará. Normalmente, si el reconocimiento falla, la aplicación incitará al usuario a repetir lo que ha dicho y el reconocimiento se intentará de nuevo. Si el sistema está correctamente diseñado y es repetidamente incapaz de entender al usuario (debido a que no se ha entendido bien la pregunta, un acento cerrado, interferencias o demasiado ruido alrededor), se retirará y por ejemplo, en un servicio de contestador automático, desviará la llamada a otro operador. La investigación muestra que las llamadas a las que se las pide replantear la pregunta o cuestión una y otra vez, en poco tiempo se frustran y se agitan.

2.2. Clasificación de los sistemas de reconocimiento de voz

Los sistemas RAH pueden **clasificarse** según los siguientes criterios:

- **Entrenabilidad**: determina si el sistema necesita un entrenamiento previo antes de empezar a usarse.
- **Dependencia del hablante**: determina si el sistema debe entrenarse para cada usuario o es independiente del hablante.
- **Continuidad**: determina si el sistema puede reconocer habla continua o el usuario debe hacer pausas entre palabra y palabra.

- **Robustez**: determina si el sistema está diseñado para usarse con señales poco ruidosas o, por el contrario, puede funcionar aceptablemente en condiciones ruidosas, ya sea ruido de fondo, ruido procedente del canal o la presencia de voces de otras personas.
- **Tamaño del dominio**: determina si el sistema está diseñado para reconocer lenguaje de un dominio reducido (unos cientos de palabras p. e. reservas de vuelos o peticiones de información meteorológica) o extenso (miles de palabras).

2.3. Problemas del reconocimiento de voz

Existen muchos factores que influyen en la dificultad del proceso de reconocimiento de voz y por tanto en su rendimiento, pero entre todos ellos destaca la **variabilidad**. La variabilidad de la señal de voz depende tanto de factores intrínsecos al fenómeno de producción de voz, como a factores externos al mismo. Dentro de los factores intrínsecos destacan los siguientes:

1. **Variabilidad de los sonidos**, debido fundamentalmente a los distintos acentos o formas de hablar de cada persona.

2. **Variabilidad en la producción de los sonidos**, debido fundamentalmente a las distintas velocidades de producción, coarticulación, inclusión de ruidos (apertura y cierre de labios, respiración, sonidos de duda, p.e., eh, uuh), condiciones acústicas (hablar en ambientes ruidosos), contexto de la conversación, estado anímico, etc. Entre los factores externos destacan:

    a. **Variabilidad en la cadena de conversión y transmisión de la señal eléctrica**, debido a las diferencias entre las características de los micrófonos, líneas telefónicas, etc.

    b. **Variabilidad en el ruido captado con la señal de voz**, debido a la existencia en las proximidades del micrófono de otras fuentes sonoras (TV, radio, carretera, impresoras, otras conversaciones, etc.)

A estos factores de variabilidad acústica habrá que añadir otros factores de variabilidad lingüística relacionados con las distintas formas dialécticas de hablar un idioma, la utilización de palabras no contempladas en el vocabulario de la aplicación, la construcción de frases no permitidas por la gramática del lenguaje, la utilización de abreviaturas, los escenarios semánticos de las palabras,

etc. Todo ello hace que el reconocimiento automático del habla por parte de una máquina no sea un problema tan trivial como a primera vista pueda parecer.

2.4. Aplicaciones más comunes del reconocimiento de voz

Aunque en teoría cualquier tarea en la que se interactúe con un ordenador puede utilizar el reconocimiento de voz, actualmente las siguientes **aplicaciones** son las más comunes:

- **Dictado automático**: El dictado automático es el uso más común de las tecnologías de reconocimiento de voz. En algunos casos, como en el dictado de recetas médicas y diagnósticos o el dictado de textos legales, se usan corpus especiales para incrementar la precisión del sistema.
- **Control por comandos**: Los sistemas de reconocimiento de habla diseñados para dar órdenes a un computador (p.e. "Abrir Firefox", "cerrar ventana") se llaman Control por comandos. Estos sistemas reconocen un vocabulario muy reducido, lo que incrementa su rendimiento.
- **Telefonía**: Algunos sistemas PBX permiten a los usuarios ejecutar comandos mediante el habla, en lugar de pulsar tonos. En muchos casos se pide al usuario que diga un número para navegar un menú.
- **Sistemas portátiles**: Los sistemas portátiles de pequeño tamaño, como los relojes o los teléfonos móviles, tienen unas restricciones muy concretas de tamaño y forma, así que el habla es una solución natural para introducir datos en estos dispositivos.
- **Sistemas diseñados para discapacitados**: Los sistemas de reconocimiento de voz pueden ser útiles para personas con discapacidades que les impidan teclear con fluidez, así como para personas con problemas auditivos, que pueden usarlos para obtener texto escrito a partir de habla. Esto permitiría, por ejemplo, que los aquejados de sordera pudieran recibir llamadas telefónicas de manera que aparezca escrito en su teléfono aquello que habla con la otra persona.

2.5. Aplicaciones en el mercado

El reconocimiento de voz es un mercado poco evolucionado debido a que la tecnología del habla es un campo difícil de desarrollar, y está aún por explotar. Sin embargo, podemos encontrar algunas compañías que desarrollan esta tecnología para todo tipo de aplicaciones, como por ejemplo las siguientes:

- **Dragon NaturallySpeaking:** *Nuance*, la empresa que gestiona esta aplicación, es una de las más avanzadas en reconocimiento de voz. Dragon es la aplicación de reconocimiento de voz más conocida del mercado. Aunque es de pago, ofrece muchas opciones al usuario que la utilice. Dragon utiliza sofisticados modelos acústicos y estadísticos que le permiten adaptarse rápidamente al usuario de distintas formas, familiarizándose con el sonido de su voz y con las palabras que utiliza al dictar.

  Dragon aprende de la voz del usuario y pronunciación a medida que lo utiliza. Para obtener los mejores resultados, es aconsejable desarrollar unos sencillos hábitos, como colocar el micrófono siempre en la misma posición, y aprovechando las sencillas herramientas que ofrece Dragon para mejorar la precisión: por ejemplo, agregar al vocabulario de Dragon su propia jerga, acrónimos y frases o palabras que requieran una pronunciación o un deletreo especial.

  Dragon permite importar listas completas de entradas de vocabulario de una vez. Asimismo, puede proporcionar a Dragon textos similares a los que va a dictar para que los "estudie" rápidamente.
- **Verbio:** El reconocimiento de voz *Verbio ASR* (Automatic Speech Recognition) es la tecnología que convierte, de forma automática, una locución de habla en texto. Básicamente permite al sistema que dispone de dicho motor "entender" o interpretar el contenido de una locución con independencia de la voz de locutor. Dispone en la actualidad de una amplia gama de aplicaciones, tanto telefónicas, como multimedia.

  El sistema reconoce las palabras o conjunto de palabras dichas entre un grupo de opciones, siendo su evolución hacia el reconocimiento de lenguaje natural *Verbio Vox Populi*. *Verbio VoxPopuli* es el motor de transcripción basado en patrones estadísticos (SLM) orientado al reconocimiento de habla espontánea en entornos de gran vocabulario.

  Las altas tasas de reconocimiento de voz de *Verbio ASR* se basan en su capacidad de adaptación en todos los entornos, no solo de las gramáticas u opciones, sino también de

la adaptación a gran cantidad de modelos acústicos existentes. Siendo además un ASR independiente del locutor. Esta tecnología es de pago.

- **TELL ME MORE:** Es un portal web que ofrece soluciones de formación para el aprendizaje de idiomas. Muchas de sus actividades funcionan con reconocimiento de voz. Este un buen ejemplo para este proyecto de actividades que el usuario ha de desarrollar utilizando el habla para mejorar su pronunciación. Por ejemplo, Tell Me More utiliza el reconocimiento de voz en actividades de diálogo de expresión y comprensión oral, o pronunciación de palabras y frases. A continuación se muestra una figura explicativa, sobre el cuadro de reconocimiento de voz que utilizan:

Figura 3.1: Cuadro de reconocimiento de voz TELL ME MORE

Dispone de dos modos de funcionamiento: el *modo manual*, en el que el usuario tiene que pulsar el botón de grabar antes de pronunciar la respuesta, y el *modo automático*, en el que el reconocimiento de voz se pone en marcha automáticamente al principio de la actividad permitiendo una conversación más fluida. Además, el usuario puede ajustar la dificultad para perfeccionar su pronunciación. Esta aplicación es de pago.

- **CMU Sphinx:** es el término general para describir un grupo de sistemas de reconocimiento de voz desarrollado en la Universidad de Carnegie Mellon. Incluye una serie de programas para reconocimiento de voz y un entrenador modelo acústico.

  Sphinx es un sistema de habla continua y reconocimiento de habla, utiliza el Modelo oculto de Márkov (HMMs) y un lenguaje de modelado estadístico de n-gramas. Sphinx interpreta voz hablada en forma continua, reconocimiento de habla de vocabulario amplio. Sphinx 4 es la última versión de Sphinx y está escrito íntegramente en lenguaje de programación Java. Este sistema de reconocimiento de voz es de libre distribución.

- **TalkingJava SDK:** es una librería desarrollada por Cloud Garden que realiza una implementación de JSAPI para la plataforma Windows. *TalkingJava SDK* es de libre distribución para un uso no comercial. Si su utilización se realiza en compañías o instituciones es necesario adquirir una licencia.

  JSAPI define una interfaz software multiplataforma estándar y fácil de usar para implementar la tecnología de habla. Sus objetivos son servir de soporte para sintetizadores de voz y sistemas de reconocimiento de habla, proveer una interfaz multiplataforma robusta para la síntesis y el reconocimiento del habla y soportar la integración con otras capacidades de la plataforma Java, como por ejemplo la API de Java Media. Además pretende ser simple, compacto y fácil de aprender y manejar.
- **Microsoft Speech SDK:** es una librería de libre distribución definida por Microsoft. Se puede utilizar para desarrollar aplicaciones de voz con Visual Basic, JavaScript y otros lenguajes de programación.
- **WebSpeech API:** es una librería de JavaScript que permite a los desarrolladores incorporar reconocimiento de voz, conversión de voz a texto y síntesis de voz en sus páginas web. Esta librería ha sido realizada por el equipo de W3C que implementa estándares de diseño web y aplicaciones, además de muchas otras aplicaciones. Hay que tener en cuenta que en este momento WebSpeech API sólo está implementado en el navegador Google Chrome para poder utilizarlo.

Hasta aquí, hemos detallado algunas de aplicaciones que utilizan reconocimiento de voz, más o menos conocidas. Con estos ejemplos hemos querido introducir al lector sobre qué posibles usos se dan reconocimiento de voz actualmente. A continuación pasaremos al diseño de la aplicación.

3. CAPÍTULO 3: DISEÑO DE LA APLICACIÓN

En este apartado, vamos a explicar los diferentes tipos de lenguajes utilizados para llevar a cabo la aplicación, así como las herramientas manejadas para la realización de la misma.

Lo primero que se hizo antes de comenzar con la realización de la aplicación, fue una búsqueda de librerías basadas en reconocimiento de voz, de forma que se adecuara a la aplicación que deseamos. En primer lugar, encontramos JSAPI (Java Speech API) ya mencionado antes en la sección 2 dedicada al reconocimiento de voz. Por ello, se intentó una prueba con un ejemplo que parecía funcionar con una implementación de JSAPI.

Después de probar con JSAPI, se decidió hacer una búsqueda más exhaustiva de reconocedores de voz de libre distribución, de los que encontramos varios, tales como Sphinx, MSAPI (Microsoft Speech API), etc.

Finalmente encontramos una solución hallada en un ejemplo de presentación de HTML5 de sus nuevas características, en el que implementaban WebSpeech, un API que consiste una caja de texto adaptada con un icono de micrófono. Al pulsar esta imagen empieza a realizar la función de reconocimiento de voz, y el usuario puede hablar por el micrófono, de modo que el reconocedor de voz traduce lo dicho por el usuario a texto, plasmándolo en una caja de texto al que está asociado el icono.

Como buscábamos una manera de implementar reconocimiento de voz en la web porque queríamos incluir nuestra aplicación en la plataforma web ILLLab, y además se encontró WebSpeech, podríamos realizar la aplicación de forma sencilla. Al disponer de los conocimientos sobre tecnologías web impartidos en la carrera, optamos por esta opción.

3.1. Intento con Java Speech, TalkingJava SDK de Cloud Garden

Se desarrolló una prueba con Java Speech API mediante un ejemplo encontrado en la web referenciada en la bibliografía. En el ejemplo se detalla la utilización de una librería de Cloud Garden, TalkingJava SDK con implementación JSAPI.

En primer lugar, se desarrolló una prueba del ejemplo tal como aparece en la página para comprobar su funcionamiento, y el ejemplo parecía funcionar correctamente aunque tenía

algunos fallos de comprensión. En el ejemplo, se definía un fichero de gramática en el que se debía escribir las palabras que se deseaban reconocer, por lo que se hizo una lista de palabras de prueba para la aplicación y las introduje en el fichero. Después de realizar el ejemplo, se implementó una adaptación a lo que sería la aplicación (parecido al juego *pasapalabra*), con una lista de palabras inicial con sus definiciones que añadimos al fichero de gramática ya mencionado para que el reconocedor de voz pudiera reconocerlas correctamente.

Aquí, el reconocedor de voz empezó a dar muchos fallos de comprensión, ya que al decir más de una sílaba en palabras de varias sílabas, escribía varias palabras que no eran la que el usuario buscaba. Además, también presentaba una posibilidad de fallo de comprensión alta con las palabras monosilábicas.

Otro problema que presentaba JAVA era su implementación en la página web, ya que se encontró la forma de implementarlo, los applet, pero pasaron a ser muy obsoletos en la web, por lo que, junto con el problema de reconocimiento de voz, se decidió buscar otros reconocedores de voz de libre distribución.

3.2. Web Speech

Después del intento fallido de JSAPI, se encontró este WebSpeech API, que es más adecuado para este proyecto, porque se quería introducir nuestra aplicación en la plataforma web ILLLab. Asimismo, hacer la aplicación en HTML5 y JavaScript, nos facilitaría la implementación de la aplicación en la web.

WebSpeech es una librería de JavaScript que permite a los desarrolladores incorporar reconocimiento de voz, conversión de voz a texto y síntesis de voz en sus páginas web. Esta librería ha sido realizada por el equipo de W3C que implementa estándares de diseño web y aplicaciones, además de muchas otras aplicaciones. Hay que tener en cuenta que en este momento WebSpeech API sólo está implementado en el navegador Google Chrome para poder utilizarlo.

En esta aplicación se utiliza un método reducido que tiene WebSpeech para su uso fácil en las páginas web. Se trata de un atributo añadido en la etiqueta input de HTML, en concreto en cajas de texto. Para implementarlo se deberá usar la siguiente sintaxis:

"<input type="text" lang="en" x-webkit-speech>"

Además de este atributo, se encuentra el atributo ***lang***, que permite elegir el idioma en el que se desea el reconocimiento de voz, ya que WebSpeech permite el reconocimiento de voz en distintos idiomas. Por defecto se corresponderá con el idioma que use el usuario en su navegador web, aunque en la aplicación definimos el inglés como idioma porque el proyecto está diseñado para que los estudiantes practiquen inglés.

Como en nuestra aplicación sólo se desea el reconocimiento de voz de una palabra, no se necesitaba más que este pequeño añadido a la caja de texto para activar el reconocimiento de voz con sólo un clic.

Hay otro método más extenso, que podría implementar mejoras a la aplicación, como por ejemplo, que la aplicación sea sólo de reconocimiento de voz, enviando la respuesta cada vez que el usuario pare de hablar y pasando a la siguiente letra, o añadiendo síntesis de voz a la aplicación, permitiendo al usuario escuchar las definiciones y deducir las respuestas sin tener que leer. Este método es usando WebSpeech API en JavaScript.

Las ventajas de usar WebSpeech API con respecto al método abreviado son las siguientes:

- La posibilidad del dictado de texto continuo, sin pausas.
- La posibilidad de implementar una multi-sesión de reconocimiento de voz y guardar el resultado.
- La posibilidad de insertar la voz reconocida en cualquier parte de un texto.
- Puede ser utilizado por cualquier elemento HTML (puedes implementar comandos de voz).
- La posibilidad de visualizar los resultados de reconocimiento provisionales.

Las desventajas de usar WebSpeech API son las siguientes:

- El usuario debe permitir el uso del micrófono antes de iniciar la sesión.
- La duración de la sesión es de un máximo de 60 segundos.

Estas desventajas son debidas a que la implementación del reconocimiento del WebSpeech API se aloja en el servidor y no localmente en el navegador. Por ello tiene un límite de tiempo en el reconocimiento. Actualmente los navegadores como Firefox o Safari no soportan WebSpeech API, excepto Google Chrome, pero se espera que pronto se implemente.

Además de lo ya mencionado, en la especificación del API se dan los siguientes casos en los que se pueden utilizar WebSpeech:

- Búsqueda en la web por voz.
- Interfaz de comandos de voz.
- Dominio gramáticas específicas en entradas anteriores.
- Dominio gramáticas específicas rellenado de varios campos de entrada.
- El reconocimiento continúo de diálogo abierto.
- Detección de actividad de voz.
- Estructura Temporal de Síntesis para proporcionar una reacción visual.
- Traducción de voz.
- Voz habilitada para cliente de correo electrónico.
- Sistemas de diálogo.
- Interacción multimodal.
- Instrucciones de voz para conducción.
- Videojuegos multimodal.
- Búsqueda multimodal.

A continuación se explicará los lenguajes utilizados para la realización del proyecto, que se dividen en lenguajes utilizados en el lado del cliente y lenguajes utilizados en el lado del servidor.

3.3. Lenguajes utilizados en el lado del cliente

Antes de nada vamos a explicar en qué consiste el *"lado del cliente"* y del *"servidor"*, y la comunicación que se realiza entre ellos para desarrollar una página dinámica.

En concreto, el lado del cliente significa que la ejecución de los programas o *scripts* se realizan en el navegador del usuario. Normalmente el navegador web es también llamado cliente web y se denomina "cliente" porque hace las tareas de solicitud y consumo de servicios. El navegador o cliente web se comunica con un servidor a través de una conexión y le solicita páginas. El servidor web le devuelve el resultado en una página HTML para que el navegador web las interprete y las muestre en la pantalla al usuario que las ha solicitado.

Los lenguajes del cliente son los que se ejecutan en el cliente web, como JavaScript, y los del lado del servidor son los que se ejecutan en el servidor web, como PHP. Un lenguaje del lado del cliente es totalmente independiente del lado del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de solicitar nada más que la página en sí al servidor Web. Inversamente, un lenguaje en el lado del servidor es independiente del cliente donde ejecuta los scripts almacenados en el servidor que es el que los ejecuta y traduce a HTML para devolvérselo al cliente, por lo que permanecen ocultos para el cliente. Este hecho puede resultar una forma legítima de proteger el trabajo intelectual realizado.

La siguiente figura nos muestra un ejemplo claro de cómo funciona:

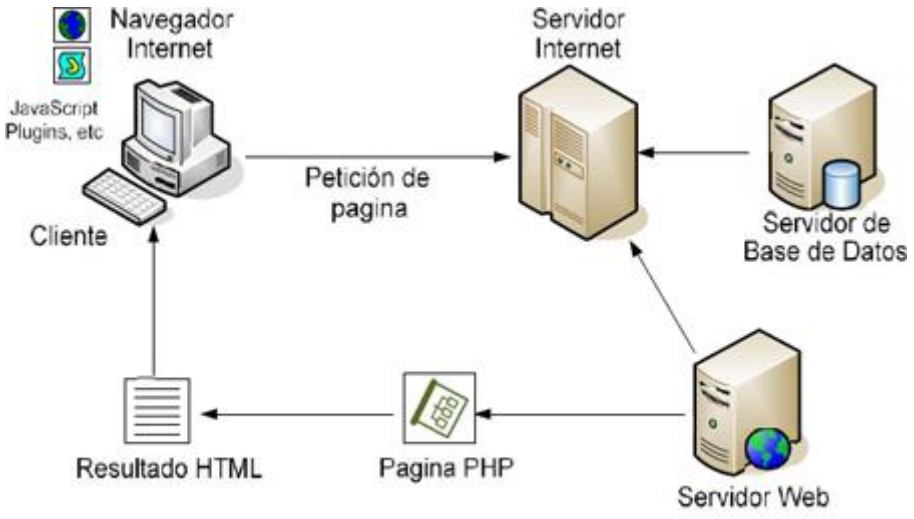


Figura 4.1 - Programación en el cliente y el servidor

Un ejemplo para la figura mostrada anterior, sería solicitar la página web donde alojamos nuestra aplicación. Esto se realiza mediante el protocolo HTTP (Hyper Text Transfer Protocol) que es un protocolo utilizado para transacciones y sigue un esquema de petición-respuesta entre el cliente y el servidor. De esta manera el servidor web devolvería como resultado la página HTML solicitada. Igualmente si la página web que solicitas es una página PHP, el servidor ejecutaría los scripts de la página PHP y que cuando finalice devolvería un resultado HTML al cliente.

Los lenguajes de programación utilizados en el cliente son:

- HTML
- JavaScript
    - Jquery
    - Ajax

3.3.1. HTML5

HTML5 es la quinta revisión de HTML (Hyper Text Markup Language), que es regulado por el *World Wide Web Consortium* (W3C) al igual que Web Speech. HTML5 todavía se encuentra en desarrollo, por ello algunas versiones de navegadores no admiten los nuevos elementos introducidos y hay que actualizarlos a la última versión, siempre y cuando los programadores de los navegadores incluyan estos cambios.

HTML es el lenguaje de marcado básico del lado del cliente utilizado por los desarrolladores para la elaboración de páginas web. Un lenguaje de marcado es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. Los lenguajes de marcado no son lenguajes de programación ya que no tienen funciones aritméticas ni variables.

Un documento HTML se escribe en forma de *etiquetas* rodeadas por corchetes (<>), describiendo hasta cierto punto la apariencia de un documento y puede incluir o hacer referencias a programas llamados *scripts*.

Las *etiquetas* son marcas insertadas en el documento HTML para proporcionar información sobre un elemento y son la estructura básica de HTML. Las etiquetas tienen dos propiedades: atributos y contenido. Cada atributo y contenido tiene ciertas restricciones para que se considere válido al documento HTML. Un elemento generalmente tiene una etiqueta de inicio (<elemento>) y una etiqueta de cierre (</elemento>). Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido está ubicado entre las dos etiquetas (por ejemplo, <elemento atributo="valor">Contenido</elemento>). Algunos elementos, tales como <br>, no tienen contenido ni llevan una etiqueta de cierre. HTML5 incorpora etiquetas nuevas como <audio> y <video>, y elimina otras que están en desuso de versiones anteriores.

Los *scripts* son programas interpretados por los navegadores web para realizar páginas web dinámicas. El lenguaje más utilizado para realizar scripts es JavaScript.

HTML5 trata de mejorar la estructura de la semántica de las etiquetas que utiliza, también introducen el uso aplicaciones web *offline* gracias a una aplicación cache de HTML5, además de implantar almacenamiento local, por bases de datos indexadas y especificaciones API de archivos. También permite el acceso a dispositivos mediante un API de geolocalización, o accesos a micrófonos/cámaras desde entradas de audio/video hasta datos locales como contactos.

HTML5 tiene mejor eficiencia de conectividad entre clientes y servidores, gracias a los WebSockets y Server-Sent Events. También incluye multimedia en sus etiquetas para el audio y video. Además, añade características para mejorar los efectos visuales como 3D, WebGL y SVG. También puedes mejorar el diseño de tu web con CSS 3, para dar todo tipo de efectos distintos a tu web.

Si se quiere ver un ejemplo de todas estas novedades de HTML5, se pueden percibir en esta presentación de diapositivas: http://slides.html5rocks.com/#landing-slide.

HTML se divide en una estructura de dos secciones: *head* y *body*. En el *head* se introduce aquellos enlaces con los archivos que componen la página web, como las CSS (Cascading Style Sheets o hojas de estilo en cascada) y los *scripts*, además de pequeñas cosas como el título de la página y metalenguajes. En el *body* se encuentra nuestra verdadera estructura de elementos de la página, como los *div* y *span*, que son bloques que colocamos en la página para darle formato a nuestro antojo y en los que incluiremos otros elementos HTML. También podemos encontrar elementos como párrafos y listas ordenadas para exponer la información que deseamos.

En la aplicación introducimos en *head* de nuestro documento HTML todas las funciones de JavaScript para hacer funcionar correctamente la aplicación, las hojas de estilo para dar formato al diseño de la web y otros añadidos como la librería para utilizar jQuery o el temporizador de cuenta atrás que también usa funciones de jQuery.

En el cuerpo metemos las imágenes de los semáforos en una lista ordenada y que se pone en horizontal junto con las letras que definen a que letra pertenece cada semáforo. Después se introduce el temporizador, y finalmente, nos encontramos con los formularios que enviarán los datos que se desean en cada caso.

Podemos encontrarnos con cuatro formularios que serán mostrados según el momento en el que se encuentre la aplicación:

- Formulario de introducción: Aquí nos encontraremos con dos botones, *instructions* y *start*. *Instructions* nos mostrará las instrucciones del juego y su funcionamiento y *start* dará comienzo al inicio del juego, enviando al servidor los datos para que los procese en consecuencia.

- Formulario del juego: En este formulario es dónde se desarrollará el *juego*, el cual muestra la letra actual de la palabra a adivinar en ese momento junto con la definición de la palabra, una caja de texto para responder y dos botones, *pass* y *submit*. *Pass* enviará al servidor que desea pasar a la siguiente letra sin resolver la actual, mientras que *submit* enviará la respuesta escrita en la caja de texto para procesar si es correcta o no y almacenar el resultado.
- Formulario de finalización: Una vez terminado el juego sustituirá al formulario anterior y dará los resultados obtenidos por el servidor web.
- Formulario de error: En caso de error este formulario aparecerá para indicarnos un error junto con un botón para volver a la pantalla de inicio.

En nuestra aplicación utilizamos HTML5 únicamente para la introducción del elemento input de WebSpeech. Los demás elementos HTML están incluidos en versiones anteriores de HTML, y que básicamente constituyen el contenido de nuestra página principal *"index.html"* de la aplicación.

3.3.2. JAVASCRIPT

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario de compilar programas para ejecutarlos. En otras palabras, los programas escritos en JavaScript se pueden probar directamente en el navegador web sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript y Java no están relacionados y tienen semánticas y propósitos diferentes. Aunque adopta nombres y convenciones del lenguaje de programación Java, se diseñó con una sintaxis similar a la de C.

JavaScript se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos, como textos que aparecen y desaparecen, animaciones, eventos que se producen al pulsar un botón o ventanas con mensajes de aviso al usuario.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del *Document Object Model* (DOM). Además de DOM utilizaremos JQuery para manipular el árbol DOM y dar efectos a la página web, y AJAX para comunicarnos asíncronamente con el servidor web. Todos estos elementos estarán descritos en los siguientes apartados.

3.3.2.1. DOM

DOM es una *interfaz de programación de aplicaciones* (API) para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como JavaScript. Por ejemplo, si queremos cambiar el contenido de un párrafo al pulsar un botón debemos acceder a él mediante herramientas de DOM, por ejemplo para cambiar su texto de forma dinámica, *“transformando”* la página original.

Para realizar estas transformaciones, DOM convierte todos los documentos HTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama "*árbol de nodos*".

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro tipos de nodos:

- ***Document*:** nodo raíz del que derivan todos los demás nodos del árbol.
- ***Element*:** representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- ***Attribute*:** se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par *atributo=valor*.
- ***Text*:** nodo que contiene el texto encerrado por una etiqueta HTML.

3.3.2.1.1. Funciones DOM utilizadas en la aplicación

Las funciones que proporciona DOM para acceder a un nodo a través de sus nodos padre consisten en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama terminada por el nodo buscado. Sin embargo, cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo y no llegar hasta él descendiendo a través de todos sus nodos padre.

Por ese motivo, no se van a presentar las funciones necesarias para el acceso jerárquico de nodos y se muestran solamente las que permiten acceder de forma directa a los nodos.

Un ejemplo de un *árbol de nodos* sería el siguiente:

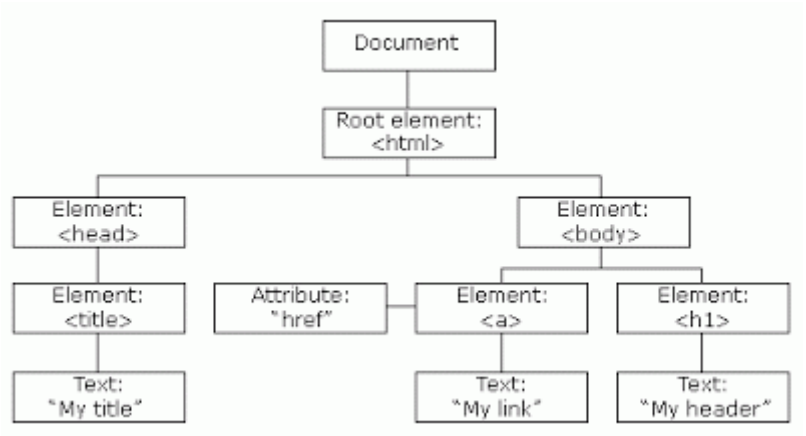


Figura 4.2: Ejemplo de árbol de nodos DOM

El documento escrito en HTML de la figura anterior quedaría de la siguiente manera:

```
<html>

<head>

<title>My title</title>

</head>

<body>

<a href="">My link</a>

<h1>My header</h1>

</body>

</html>
```

Para acceder a estos elementos del árbol de nodos, DOM proporciona una serie de funciones muy útiles, pero en este documento sólo explicaremos las utilizadas para la realización de la aplicación.

Las funciones DOM que usaremos son las siguientes:

- *document.createElement(etiqueta HTML):* está función sirve para crear un elemento dentro del árbol de nodos del documento HTML, que será una etiqueta pasada como parámetro. Después de explicar la siguiente función pondremos un ejemplo donde la utilizamos en la aplicación.
- *document.getElementById(identificador):* esta función sirve para obtener un elemento del árbol de nodos del documento HTML cuyo atributo *id* coincide con el identificador pasado como parámetro, de forma que devuelve únicamente el nodo deseado. Esta función es de las más utilizadas cuando se desarrollan aplicaciones web dinámicas.

En la aplicación la función *document.createElement()* es utilizada junto con la función *document.getElementById()* para comprobar que se está utilizando Google Chrome y poder utilizar el complemento de reconocimiento de voz añadido, definiendo el siguiente fragmento de código:

```
31      if (document.createElement("input").webkitSpeech === undefined) {
32          document.getElementById("browser").innerHTML += "<p style='te
33      }else{
34          document.getElementById("browser").innerHTML += "<p style='te
35      }
```

Antes de explicar este fragmento, en él podemos observar que utilizamos una propiedad llamada *innerHTML.* Esta propiedad establece o devuelve el contenido de un elemento HTML, es decir, el texto escrito del elemento. Por lo que junto con la función *document.getElementById()*, estableceremos el texto que estará escrito en un elemento HTML con un identificador denominado "browser".

El fragmento de código describe que si se da la condición de que en el documento HTML se ha creado un elemento *"input"* con un atributo *webkitSpeech* sea indefinido (hacemos esta comparación porque sólo en Google Chrome podrá ser definido y en los demás navegadores todavía no está soportado), entonces la aplicación escribirá el contenido de un elemento HTML con identificador *"browser".* En concreto escribirá un párrafo que indicará que el usuario no está utilizando Google Chrome y por tanto no podrá optar a la opción de reconocimiento de voz. En caso contrario, la aplicación escribirá en el elemento HTML con ese mismo identificador, *"browser",* un párrafo indicando que el usuario si lo está usando y podrá acceder al reconocimiento de voz.

3.3.2.2. JQUERY

JQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML y sus elementos, manipular el árbol de nodos DOM, manejar eventos, desarrollar animaciones y agregar la técnica AJAX a las páginas web. JQuery es de software libre y código abierto, lo que significa que cualquier desarrollador puede utilizarlo de forma gratuita y consultar su código en cualquier momento sin ninguna censura.

Las características principales de lo que JQuery nos puede ofrecer son las siguientes:

- Interactividad y modificaciones del árbol de nodos DOM.
- Eventos.
- Manipulación de la hoja de estilos CSS.

- Efectos y animaciones.
- AJAX.
- Soporta extensiones.
- Diferentes utilidades como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.
- Es compatible con los navegadores Mozilla Firefox 2.0, Internet Explorer 6, Safari 3, Opera 10.6, Google Chrome 8 y sus versiones posteriores.

3.3.2.2.1. Funciones jQuery utilizadas en la aplicación

La característica principal de esta biblioteca JavaScript es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. La forma de interactuar con la página es mediante la función $(), un alias de jQuery(), que recibe como parámetro una expresión CSS (hoja de estilo) o el nombre de una etiqueta HTML y devuelve todos los nodos (elementos HTML) que concuerden con la expresión. Esta expresión es denominada **selector** en la terminología de jQuery. Por ejemplo:

La expresión *$("#identificador");* devolverá el elemento que tenga el atributo id="identificador". El carácter "#" se refiere al identificador. Los identificadores son únicos, por lo que siempre se referirá a un único elemento HTML.

La expresión *$(".clase");* Devolverá una matriz de elementos con el atributo class="clase". El carácter "." hace referencia a una clase que pueden usar varios elementos.

Antes de realizar cualquier acción en el documento con jQuery, debemos esperar a que el documento HTML esté cargado completamente para que no surjan errores en las modificaciones del árbol de nodos DOM. Para ello usamos la función *$(document).ready();* de esta forma:

```
2 $(document).ready(function() {
3     //Aquí van todas las acciones del documento.
4 });
```

Ahora que sabemos cómo funciona básicamente jQuery, vamos a proceder a las funciones que utilizamos en la aplicación para después entender cómo funciona nuestra aplicación. Estas funciones las podemos encontrar en el API de jQuery en el siguiente enlace http://api.jquery.com/. Además de explicarlas, añadiremos un ejemplo utilizado en la aplicación para una mejor comprensión:

- **.show():** muestra los elementos seleccionados que están ocultos.

```
118     $("#way").show();
```

Mostramos el elemento cuyo identificador seleccionado es "*way*".

- **.hide():** oculta los elementos seleccionados anteriormente por el selector.

```
2     $("#trafficLight").hide();
```

En esta línea de código oculta el elemento con identificador "*trafficLight*" seleccionado por la función $(). En este caso escondemos el formulario con ese identificador porque no queremos que aparezca en ese momento.

- **.toggle():** muestra u oculta los elementos relacionados con el selector.

```
76     $("#instructions").toggle();
```

Cada vez que se ejecuta esta función alterna entre mostrar u ocultar el elemento cuyo identificador es "*instructions*", por ejemplo, si se encuentra oculto, entonces se mostrará y viceversa.

- **.fadeIn():** muestra los elementos seleccionados de estar ocultos a opaco con un fundido.

```
88     $("#end").fadeIn(1000);
```

Se utiliza para mostrar el elemento HTML cuyo identificador es "*end*", con un efecto de fundido. El número 1000 que aparece como parámetro de la función, es para determinar la duración de la animación y estará expresado en milisegundos.

- **.click():** enlaza un manejador de eventos al pulsar un "clic" en un elemento HTML o al activar este evento en un elemento. Con manejador de eventos nos referimos a una función que actué cuando se active el evento.

```
10      $('#start').click(start);
```

La función que manejará el evento será *start*, que será pasada como parámetro dentro de *click*, que está asociada con el elemento HTML cuyo identificador es *"start"* (en este caso, el manejador del evento y el identificador del elemento HTML es el mismo).

- **.keypress():** enlaza un manejador de eventos al pulsar una tecla del teclado o al activar este evento en un elemento HTML.

```
2 $('input#wordSpeech').keypress(function(e){
3     if(e.which == 13){
4         return false;
5     }
6 });
```

En este caso utilizamos esta función para deshabilitar la tecla *"intro"* de nuestro formulario de juego, para que al capturar este evento no realice ninguna acción, ya que por defecto el formulario se enviaría al servidor.

- **.focus():** enfoca o fija el cursor un elemento HTML para hacerlo notar al usuario.

```
141             $("#wordSpeech").focus();
```

Enfocamos el elemento cuyo identificador es *"wordSpeech"*. Esta función la utilizamos para enfocar la caja de texto donde escribimos las palabras cada vez que enviamos una palabra, para que el usuario pueda rápidamente escribirla sin tener que volver a seleccionar la caja de texto.

- **.attr():** obtiene o modifica el valor de un atributo del elemento seleccionado.

```
99          $(tfLetter).attr("src","img/small-rojo.png");
```

Aquí queremos modificar el valor del atributo *"src"* del elemento HTML seleccionado, modificándolo por el valor *"img/small-rojo.png"*. El atributo *"src"* significa source, se refiere a la fuente donde podemos encontrar la imagen que queremos cambiar.

- **.removeAttr():** elimina un atributo de los elementos seleccionados.

```
142            $('#start').removeAttr("disabled");
```

Elimina el atributo *"disabled"* del elemento cuyo identificador es "*start*". Se utiliza para volver a habilitar el botón *start*.

- **.val():** obtiene el valor actual del primer elemento de los seleccionados.

```
121    var start = $("#start").val();
```

Obtenemos el valor del elemento HTML cuyo identificador es "*start*" y lo guardamos en la variable *start*.

- **.html():** obtiene o modifica el contenido HTML del primer elemento de los seleccionados.

```
132            $('#letter').html(data.letter);
```

Modificamos el contenido HTML del elemento cuyo identificador es "*letter*" por el de la variable *data.letter*.

3.3.2.2.2. Temporizador de cuenta atrás

Una parte importante de nuestra aplicación es el temporizador de cuenta atrás. Buscando la forma de cómo realizar uno, se encontró una página web en la que un programador desarrolló en jQuery un plugin propio que contenía contadores con múltiples opciones, diferentes formas y muchas más características. Este plugin nos permite mostrar un temporizador implementándolo en una etiqueta HTML de bloque (o) para situarlo dónde se desee. La página web del temporizador es http://keith-wood.name/countdown.html, en la que podemos encontrar el API con sus funciones y variables.

Este temporizador es perfecto para la aplicación que estamos realizando, y además de ser de código abierto y software libre, no tenemos ninguna restricción a la hora de su uso, por lo que lo implementamos en el juego.

En el siguiente trozo de código se muestra la creación del temporizador:

```
15 var shortly = 0;
16 $('#defaultCountdown').countdown({until: shortly,
17                                   format: 'MS',
18                                   onExpiry: end});
```

Creamos el temporizador en un elemento HTML cuyo identificador es "*defaultCountdown*" que corresponde a una etiqueta * * con ningún contenido. Proseguimos con la llamada a la función *countdown* que creará el temporizador con los siguientes parámetros:

- ***until:*** esta opción determina el tiempo del temporizador, en este caso será el valor de la variable *shortly* que es igual a cero.
- ***format:*** esta opción determina el formato con el que queremos mostrar nuestro temporizador en la aplicación. Utilizaremos el formato *'MS'* que significa que mostraremos sólo los minutos y segundos.
- ***onExpiry:*** está opción es una función que establece que al finalizar el temporizador ejecutará la función que determinemos. Aquí ejecutaremos la función *end* de nuestro código JavaScript para finalizar la partida del juego.

A continuación explicaremos las opciones que utilizamos en la aplicación, aunque son pocas, se pueden realizar muchas otras funciones con este temporizador.

- ***getTimes:*** Devuelve una matriz con el conjunto de periodos actuales para el contador seleccionado. Las entradas de la matriz están determinadas en el siguiente orden: años, meses, días, horas, minutos y segundos.

```
2 $('#defaultCountdown').countdown('getTimes');
```

- ***option:*** define una serie de opciones para el temporizador. Todas las opciones se pueden encontrar en la documentación de este plugin.

```
6 $('#defaultCountdown').countdown('option', {until:suma});
```

Para este caso, definimos que el valor del temporizador ahora es establecido por el valor de la variable suma.

- ***destroy:*** "destuye" el temporizador, es decir, retira la funcionalidad del temporizador dado a la etiqueta HTML a la que estaba asociado.

```
8 $('#defaultCountdown').countdown('destroy');
```

- ***periodsToSeconds:*** Esta función convierte una matriz de periodos a su número equivalente en segundos. El parámetro *periods* debe ser una matriz como la descrita en la función *getTimes*.

```
4 $.countdown.periodsToSeconds(periods);
```

3.3.2.3. AJAX

AJAX es el acrónimo de *Asynchronous JavaScript And XML*. AJAX es una técnica desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador web de los usuarios, mientras que se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML (eXtensible Markup Language, no explicaremos este lenguaje ya que no se utiliza explícitamente en este proyecto).

En la aplicación se envían los datos recogidos hacia el servidor desde los formularios del juego, y queremos que el temporizador con cuenta atrás no se vea afectado por ello. Esto no era posible debido a que la comunicación era directa con el servidor. Por la necesidad en nuestra aplicación, de que la comunicación con el servidor fuera asíncrona, es decir, sin que la página se recargue, la conexión del cliente con el servidor debería ser en segundo plano. AJAX cubre esta necesidad fundamental para este proyecto, por esta razón se decidió el uso de esta tecnología web.

Todos los navegadores mencionados anteriormente en las características de jQuery soportan la tecnología de AJAX. A continuación procederemos a explicar la función que utilizamos en nuestra aplicación para realizar la comunicación asíncrona entre el cliente y el servidor.

```
2  $.ajax({
3      url: 'trafficLights.php',
4      type: 'POST',
5      data: dataString,
6      dataType: "json",
7      success: function(data){
8          //Aquí van las acciones en caso de éxito
9      },
10     error: function(msg) {
11         //Aquí van las acciones en caso de fallo
12     }
13 });
```

Nota: AJAX está incluido en jQuery por lo que esta función está desglosada en el API de jQuery.

$.ajax() realiza una conexión asíncrona HTTP, en la que se establecen una serie de parámetros para enviar los datos que deseamos y de esta manera recibir una respuesta HTTP por parte del servidor. Los parámetros son los siguientes:

- ***url:*** es la dirección web donde se encuentra el destino de esta solicitud HTTP, en este ejemplo se trata de nuestra página "*trafficLights.php*".
- ***type:*** es el tipo de solicitud HTTP que queremos enviar, por defecto es GET, pero como queremos que se envié POST, pues definimos este parámetro. La url de una solicitud HTTP está definida de la siguiente manera: *url?dato1=valor1&dato2=valor2* y así sucesivamente. La única diferencia que hay entre GET y POST, es que una solicitud HTTP en GET muestra los parámetros enviados en la url y POST no los muestra de modo que el usuario no pueda verlos.
- ***data:*** son los datos que queremos enviar, los enviamos como una variable de cadena de caracteres con el formato que debería enviarse en una solicitud HTTP explicado anteriormente.
- ***dataType:*** es el tipo de datos que la función espera que sea devuelta por el servidor, definimos JSON como opción porque es un formato más ligero y simple de codificar los datos.
- ***success:*** en caso de que se haya recibido respuesta del servidor, se ejecutará esta función con el código correspondiente.
- ***error:*** en caso de falle la conexión por cualquier razón, se llamará esta función ejecutando las acciones pertinentes.

3.3.2.4. Aplicación

Antes de explicar el fichero *"functions.js"*, desglosaremos la cabecera de nuestro documento HTML *"index.html"* que es importante para poder incluir los ficheros JavaScript y las hojas de estilo en nuestra aplicación.

Para incluir JavaScript en el documento HTML se puede hacer de dos maneras:

1. Incluir JavaScript en el mismo documento HTML, utilizando etiquetas <script></script> cuyo contenido sería JavaScript. Cómo hemos escogido la segunda forma, ésta no la explicaremos.
2. Definir JavaScript en un archivo externo. Definimos el archivo *"functions.js"* donde incluiremos todo el contenido JavaScript de nuestra aplicación.

   Para incluir todo el contenido JavaScript en nuestro documento HTML introduciremos en la etiqueta <head></head> las siguientes líneas de código:

```
<script type="text/javascript"
src="http://code.jquery.com/jquery-latest.js"></script>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js">
</script>
<script type="text/javascript" src="js/jquery.countdown.js"></script>
<script type="text/javascript" src="js/functions.js"></script>
```

   En las etiquetas *script* utilizaremos los atributos "*type*", que indica el tipo de código que se incluirá dentro del script, en este caso JavaScript, y "*src*" que contendrá el fichero de código. En la primera etiqueta *script* incluimos el fichero con las funciones que utiliza jQuery. En el segundo se incluyen las librerías de AJAX. En el tercero con las del temporizador de cuenta atrás, y por último incluiremos nuestras funciones de la aplicación.

Además de estos archivos, también meteremos los ficheros con las hojas de estilo (CSS) que utiliza nuestro documento HTML para darle forma y color a la página web:

```
<link rel="stylesheet" type="text/css" href="css/style.css" />
<link rel="stylesheet" type="text/css" href="css/jquery.countdown.css" />
```

Estos archivos al no ser *scripts* se enlazan con el documento HTML con la etiqueta *link*. Utilizamos los atributos "*rel*" y "*type*" para definir que son hojas de estilo y "*href*" para referenciar donde se encuentran.

Y ahora procederemos a explicar el fichero "*functions.js*", que realiza importantes funciones desde la parte del cliente para conectarse con el servidor de forma asíncrona y así poder intercambiar peticiones y respuestas HTTP transmitiendo los datos deseados sin que se recargue la página web.

"functions.js"

Nota: Cómo se han explicado anteriormente las funciones de las librerías de JavaScript, procederemos a explicar directamente el significado de cada parte de código expuesto.

Empezaremos con la función ***.ready*** ya explicada en jQuery, que sirve para realizar las siguientes acciones después de que se cargue el documento HTML completo en el navegador.

```
1 $(document).ready(function (){
```

Una vez que se cargue todo el documento en el navegador web del usuario, procederemos a esconder los siguientes elementos del documento HTML que no queremos que este a la vista del usuario.

```
2     $("#trafficLight").hide();
3     $("#end").hide();
4     $("#instructions").hide();
5     $("#way").hide();
6     $("#defaultCountdown").hide();
7     $("#error").hide();
```

Estos elementos son todos los formularios excepto el de inicio, que es el que queremos que se muestre antes de empezar la aplicación, además ocultaremos también la fila de semáforos y el temporizador.

Después proseguiremos asignando a los botones que tenemos en el juego con sus manejadores de eventos correspondientes. En este caso el evento es hacer un "click" en el botón para ejecutar las funciones correspondientes, que explicaremos más adelante.

```
9     $("#instruct").click(instructions);
10    $('#start').click(start);
11    $('#pass').click(pass);
12    $('#submit').click(submit);
```

A continuación crearemos nuestro temporizador asignándolo a un , es decir a una etiqueta de bloque con identificador "*defaultCountdown"*. Le daremos un valor de cero segundos, porque el juego no empieza hasta que se pulse el botón *start.* Además le fijaremos un formato de minutos y segundos, y cuando termine el temporizador ejecutará la función *end*, para finalizar correctamente el juego.

```
15 var shortly = 0;
16 $('#defaultCountdown').countdown({until: shortly, format: 'MS', onExpiry: end});
```

Aquí desactivaremos la tecla *Intro* del teclado para que no envíe el formulario cada vez que se pulse. Realizamos esta acción porque al enviar el formulario pulsando la tecla *Intro* no utilizaba AJAX, sino que se comunicaba con el servidor síncronamente, haciendo que nuestro temporizador se reiniciará. Igualmente le asignamos la función *submit*¸ que se encarga de enviar los datos del formulario con AJAX, pero también daba problemas porque si el usuario pulsaba *Intro* varias veces seguidas se enviaba el formulario ese mismo número de veces, y desestabilizada el orden del juego. Por lo tanto decidimos desactivarlo para que no hubiera problemas.

```
22 $('input#wordSpeech').keypress(function(e){
23     if(e.which == 13){
24         return false;
25     }
26 });
```

Este fragmento de código sirve para comprobar si tu navegador admite reconocimiento de voz, sólo soportado en este momento por Google Chrome.

```
30 if (document.createElement("input").webkitSpeech === undefined) {
31     document.getElementById("browser").innerHTML += "<p style='te
32 }else{
33     document.getElementById("browser").innerHTML += "<p style='te
34 }
```

De este modo finaliza la parte de código que se ejecutará al terminar de cargarse el documento HTML en el navegador web del usuario. Ahora explicaremos las funciones auxiliares utilizadas en las funciones principales para agilizar el código.

La función ***addSec*** añadirá 5 segundos al tiempo actual que tenga el temporizador. Para realizarlo se utilizará el objeto *Date*. El objeto *Date* contiene el formato de una fecha predeterminado, y que nuestro temporizador utiliza como formato para expresar el tiempo. Además utilizamos la función *setSeconds* del objeto *Date*, para asignarle los segundos actuales del temporizador más los 5 segundos que queremos añadir. Finalmente se le asigna este cambio al temporizador con la opción *'option'*.

```
44 function addSec(){
45     var periods = $('#defaultCountdown').countdown('getTimes');
46     var secActual = $.countdown.periodsToSeconds(periods);
47 
48     var suma = new Date();
49     suma.setSeconds(suma.getSeconds() + secActual + 5);
50     $('#defaultCountdown').countdown('option', {until:suma});
51 }
```

La función ***subSec*** restará 5 segundos al temporizador, al igual que antes se los sumaba. En esta función hemos añadido una condición, en la que si la resta de los 5 segundos con el tiempo actual del temporizador es menor que cero, queremos que se ejecute la función end para terminar correctamente con la aplicación.

```
58 function subSec(){
59     var periods = $('#defaultCountdown').countdown('getTimes');
60     var secActual = $.countdown.periodsToSeconds(periods);
61 
62     var resta = new Date();
63     resta.setSeconds(resta.getSeconds() + secActual - 5);
64     $('#defaultCountdown').countdown('option', {until:resta});
65     if((resta.getSeconds() + secActual - 5)<=0){
66         end();
67     }
68 }
```

La función ***instructions*** es el manejador de eventos cuando se pulsa el botón de *instructions*, lo único que realiza es alternar entre mostrar u ocultar la información de instrucciones para que el usuario la lea.

```
75 function instructions(){
76     $("#instructions").toggle();
77 }
```

La función ***divEnd*** sirve para ocultar el resto de elementos del juego, mostrando únicamente los resultados que el usuario ha conseguido al terminar el juego.

```
84 function divEnd(countGREEN, countRED){
85 document.getElementById("green").innerHTML="You guessed "+countGREEN+".";
86 document.getElementById("red").innerHTML="You missed "+countRED+".";
87 $("#trafficLight").hide();
88 $("#end").fadeIn(1000);
89 }
```

La última función auxiliar es ***changeTF*** que cambia la imagen de semáforo según si la respuesta del usuario es correcta, incorrecta o éste ha pasado a la siguiente letra, pareciendo tener un efecto en el que se enciende una luz del semáforo u otra, respectivamente.

```
 96 function changeTF(state, letter){
 97     var tfLetter = "#tf" + letter;
 98     if(state==0){ //RED
 99         $(tfLetter).attr("src","img/small-rojo.png");
100     }
101     if(state==1){ //GREEN
102         $(tfLetter).attr("src","img/small-verde.png");
103     }
104     if(state==2){ //YELLOW
105         $(tfLetter).attr("src","img/small-ambar.png");
106     }
107 }
```

Terminaremos explicando las funciones principales y más importantes de esta aplicación en el lado del cliente.

La función ***start*** es el manejador de eventos del botón *start*, esto es, que cada vez que el usuario pulse el botón *start*, se ejecutará esta función. Hay que pensar que esta función se ejecuta el tiempo en que tarde en dar una respuesta el servidor, ya que utilizaremos AJAX asíncronamente, por lo que si el servidor es lento es posible que tarde unos segundos. Si no recibe respuesta en un tiempo determinado, se ejecutará la función error informando de ello.

Comienza desactivando el botón *start* para que el usuario no pueda enviar más de una vez el mismo formulario y no generar problemas. Además mostraremos el formulario del juego y la fila de semáforos, y ocultaremos el título para dejar más espacio para el juego. Obtendremos el valor del botón *start* y se lo asignaremos a la cadena de caracteres que enviaremos al servidor con la solicitud HTTP de tipo POST.

Utilizaremos la función de AJAX para enviar la solicitud HTTP de tipo POST al servidor, concretamente a la página "*trafficLights.php*" esperando una respuesta codificada en JSON. En caso de éxito, se ejecutará la función *success* y en caso de error por cualquier causa, ejecutaremos la función *error* que mostrará un mensaje de error al usuario.

La función *success* obtendrá los datos de la letra y la definición con la que empezará, y los asignará a los elementos HTML correspondientes para que el usuario pueda verlos. Después activará el temporizador con cuenta atrás con un tiempo por defecto de 300 segundos y un formato de minutos y segundos, como vemos en la siguiente línea de código.

```
135 $('#defaultCountdown').countdown('option', {until: shortly, format: 'MS'});
```

```
function start() {
    $('#start').attr("disabled", "disabled");
    $("#way").show();
    $("#defaultCountdown").show();
    $("#title").hide();
    var start = $("#start").val();
    var dataString = 'start=' + start;

    $.ajax({
        url: 'trafficLights.php',
        type: 'POST',
        data: dataString,
        dataType: "json",
        success: function(data){
            $('#letter').html(data.letter);
            $('#def').html(data.definition);

            shortly = new Date();
            shortly.setSeconds(shortly.getSeconds() + 300);
            $('#defaultCountdown').countdown('option', {unt

            $("#startMenu").hide();
            $("#trafficLight").show();
            $("#wordSpeech").focus();
            $('#start').removeAttr("disabled");
        },
        error: function(msg) {
            $('#start').removeAttr("disabled");
            alert(msg);
        }
    });
}
```

Por último oculta el formulario de inicio y muestra el formulario del juego. Se fija la caja de texto donde el usuario podrá escribir la palabra sin tener que pulsar con el ratón para seleccionarla y se activa otra vez el botón *start* (aunque ya no se puede utilizar porque está oculto). De esta manera el usuario empezará con el juego.

La función ***submit*** es el manejador de eventos del botón *submit*, por lo que cada vez que el usuario pulse el botón *submit*, se ejecutará esta función.

```
function submit() {
    $('#pass').attr("disabled", "disabled");
    $('#submit').attr("disabled", "disabled");
    var letter = document.getElementById("letter").innerHTML;
    var wordSpeech = $("#wordSpeech").val().toLowerCase();
    var dataString = 'letter=' + letter + '&wordSpeech=' + wordSpeech;
```

Deshabilitará los botones *pass* y *submit* para evitar volver a enviar solicitudes al servidor, como ya hemos explicado anteriormente. Introduciremos en la variable *dataString* la cadena de caracteres con los datos que queremos enviar, en este caso serán la letra y la palabra escrita en la caja de texto.

```
164 $.ajax({
165     url: 'trafficLights.php',
166     type: 'POST',
167     data: dataString,
168     dataType: "json",
169     success: function(data){
170         $('#pass').removeAttr("disabled");
171         $('#submit').removeAttr("disabled");
172         if(data.end=="end"){
173             divEnd(data.countGREEN, data.countRED);
174             var periods = $('#defaultCountdown').countdown('getTimes');
175             if(periods[5]==null && periods[6]==0){
176                 document.getElementById("time").innerHTML = "You finish
177             }else{
178                 document.getElementById("time").innerHTML = "You finish
179             }
180             $('#defaultCountdown').countdown('destroy');
181             $('#defaultCountdown').hide();
182             changeTF(data.state, data.letterlast);
183         }else{
184             document.getElementById("letter").innerHTML = data.letter;
185             document.getElementById("def").innerHTML = data.definition;
186             $("#wordSpeech").val("");
187             $("#wordSpeech").focus();
188             if(data.state==1){
189                 addSec();
190             }else{
191                 subSec();
192             }
193             changeTF(data.state, data.letterlast);
194         }
195     },
```

Ejecutaremos la función AJAX para enviar los datos al servidor web, en concreto a la página *"trafficLights.php"*, si tiene éxito, se ejecutará la función *success* mostrada en la figura anterior. Volveremos a habilitar los botones *pass* y *submit*, y en caso de que el juego termine porque se han contestado todas las palabras de la lista de palabras generada de la base de datos, se procederá a mostrar los resultados en el formulario de resultados y ocultar el formulario de juego, destruiremos el temporizador, y por último, cambiaremos el color del último semáforo contestado. Si el juego no ha terminado, entonces sólo asignaremos a los elementos HTML la letra y la definición siguiente de la lista de palabras que recibiremos del servidor y fijaremos el cursor en la caja de texto.

Por último, en caso de que el usuario haya acertado la palabra, se añadirán 5 segundos al temporizador. Si el usuario responde incorrectamente a la palabra, entonces se restarán 5 segundos al temporizador. En cualquier caso se cambiará la imagen del semáforo al color que le corresponda (verde si es correcto, y rojo si es incorrecto).

La función de *error* vuelve a habilitar los botones *pass* y *submit*, oculta los demás elementos HTML para mostrar el formulario de error con el mensaje de error correspondiente.

```
196     error: function(msg) {
197         $('#pass').attr("disabled", "");
198         $('#submit').attr("disabled", "");
199         $("#trafficLight").hide();
200         $("#way").hide();
201         $('#defaultCountdown').countdown('destroy');
202         $("#defaultCountdown").hide();
203         document.getElementById("msgError").innerHTML = "An error
204         $('#error').show();
205     }
```

La función ***pass*** es el manejador de eventos para el botón *pass*, y funciona igual que los anteriores manejadores. Igualmente deshabilita también los botones *pass* y *submit* y asigna a la cadena de caracteres *dataString* los datos que se desean enviar, en este caso indica la letra y que se ha pulsado *pass*.

```
217 function pass() {
218     $('#pass').attr("disabled", "disabled");
219     $('#submit').attr("disabled", "disabled");
220     var letter = document.getElementById("letter").innerHTML;
221     var dataString = 'letter=' + letter + '&pass=pass';
222 
223     $.ajax({
224         url: 'trafficLights.php',
225         type: 'POST',
226         data: dataString,
227         dataType: "json",
228         success: function(data){
229             $('#pass').removeAttr("disabled");
230             $('#submit').removeAttr("disabled");
231             document.getElementById("letter").innerHTML = data.letter;
232             document.getElementById("def").innerHTML = data.definition;
233             $("#wordSpeech").val("");
234             $("#wordSpeech").focus();
235             changeTF(data.state, data.letterlast);
236         },
237         error: function(msg) {
238             $('#pass').attr("disabled", "");
239             $('#submit').attr("disabled", "");
240             $("#trafficLight").hide();
241             $("#way").hide();
242             $('#defaultCountdown').countdown('destroy');
243             $("#defaultCountdown").hide();
244             document.getElementById("msgError").innerHTML = "An error ha
245             $('#error').show();
246         }
247     });
248 }
```

Utilizando AJAX envía los datos en una solicitud HTTP de tipo POST para que el servidor los procese y devuelva una respuesta al cliente. Si recibe respuesta y todo es correcto, ejecutará la función *success* que volvería a activar los botones, asignaría a los elementos HTML, la letra y la definición de la siguiente letra del abecedario, fijaría el cursor en la caja de texto y cambiaría el color del semáforo a ámbar, ya que hemos pasado de letra y podremos volver a esta palabra en la siguiente ronda. En caso de *error*, se ocultaría todo para mostrar el formulario de error con el error que haya ocurrido y destruiría el temporizador.

La función ***end*** se encarga de avisar al servidor que el juego ha finalizado debido a la terminación del temporizador. Para ello desactiva los botones del formulario, como en las anteriores funciones, para que el usuario no pueda enviar nada, y se envían los datos de la letra actual en la que se encuentra el juego en ese momento y el parámetro *end*, para avisar que ha terminado el juego por la finalización del temporizador.

```
258 function end() {
259     $('#pass').attr("disabled", "disabled");
260     $('#submit').attr("disabled", "disabled");
261     var letter = document.getElementById("letter").innerHTML;
262     dataString = 'letter=' + letter + '&end=end';
263 
264 
265     $.ajax({
266         url: 'trafficLights.php',
267         type: 'POST',
268         data: dataString,
269         dataType: "json",
270         success: function(data){
271             divEnd(data.countGREEN, data.countRED);
272             document.getElementById("time").innerHTML = "You
273             $('#defaultCountdown').countdown('destroy');
274             $('#defaultCountdown').hide();
275             $('#pass').attr("disabled", "");
276             $('#submit').attr("disabled", "");
277         },
278         error: function() {
279             $('#pass').attr("disabled", "");
280             $('#submit').attr("disabled", "");
281             $("#trafficLight").hide();
282             $("#way").hide();
283             $('#defaultCountdown').countdown('destroy');
284             $("#defaultCountdown").hide();
285             document.getElementById("msgError").innerHTML = "
286             $('#error').show();
287         }
288     });
289 }
```

Una vez recibida la respuesta de la solicitud HTTP enviada por AJAX, si todo ha salido correctamente, se ocultarán los demás elementos HTML y se mostrará el formulario de resultados con los datos correspondientes de la partida que el usuario ha jugado. Por último se destruirá el temporizador y se habilitarán los botones otra vez. En caso de *error*, se procederá como en los casos anteriores.

3.4. Lenguajes utilizados en el lado del servidor

Como ya explicamos en el lado del cliente, el lado del servidor se encarga de proveer de recursos o servicios a los clientes, como ejecutar scripts para devolverlos en formato HTML al cliente. Los lenguajes del lado del servidor utilizados para la aplicación son los siguientes:

- PHP
- MySQL (El lenguaje que se utiliza es SQL, MySQL gestiona la base de datos)

3.4.1. PHP

PHP (Hypertext PreProcesor) es un lenguaje de programación de código en el lado del servidor desarrollado para la web y nos permite embeber su dentro de una página HTML, pudiendo así combinar ambos lenguajes. Es el lenguaje más extendido en la web para los desarrolladores, debido a su potencia y simplicidad que lo caracterizan, así como su soporte en la mayoría de los servidores de hosting. Para su uso en local se necesitará un servidor como apache para hacerlo funcionar, o también se podrá hacer uso en remoto en los servicios de alojamiento web que soporten PHP. Asimismo ofrece un sinfín de funciones para la explotación de bases de datos sin complicaciones, de sencilla manera.

En la aplicación es utilizado para el procesamiento de los datos enviados de la aplicación por el cliente y para el acceso a la base de datos MYSQL. En este caso, los datos enviados por los formularios del juego que se explicarán posteriormente. Además, se utilizan sesiones para poder almacenar variables entre letra y letra, como por ejemplo la lista de palabras escogida aleatoriamente al empezar la aplicación de la base de datos.

3.4.1.1. Sesiones PHP

Empezaremos explicando que es una cookie para entender mejor el funcionamiento de las sesiones. Una cookie es un fragmento de información que un navegador web almacena en el disco duro del visitante a una página web. La información se almacena a petición del servidor web, ya sea directamente desde la propia página web con JavaScript o desde el servidor web mediante las cabeceras HTTP, que pueden ser generadas desde un lenguaje de web como PHP. La información almacenada en una cookie puede ser recuperada por el servidor web en posteriores visitas a la misma página web.

Una sesión es un mecanismo de programación de las tecnologías de web que permite conservar información sobre un usuario al pasar de una página a otra. A diferencia de una cookie, los datos asociados a una sesión se almacenan en el servidor y nunca en el cliente.

En la mayoría de las tecnologías web, las sesiones se implementan mediante una cookie que almacena un valor que identifica al usuario en el servidor web cada vez que pasa de una página

web a otra. En el servidor web están almacenados todos los datos de la sesión y se accede a ellos cada vez que se pasa de página gracias al identificador almacenado en la cookie.

Utilizaremos las sesiones PHP en nuestra aplicación para almacenar variables que usaremos a lo largo de una partida del juego. Las variables que guardaremos son la lista de palabras aleatorias obtenida de la base de datos, un contador que indica el número de palabras que faltan por contestar, y por último una variable que almacena el número de ronda por la que el usuario se encuentra en ese momento.

Las funciones aplicadas en el juego sobre sesiones PHP y que ahora vamos a describir son las siguientes:

- ***session_start():*** crea una sesión o reanuda la actual basada en un identificar de sesión pasado mediante una cookie. Hay que tener en cuenta que esta función es llamada en toda página php para utilizar las variables almacenadas en la sesión.
- ***session_unset():*** libera todas las variables de sesión actualmente registradas.
- ***session_write_close():*** finaliza la sesión actual y almacena la información de la sesión.
- ***session_destroy():*** destruye toda la información almacenada con la sesión actual, pero no destruye ninguna de las variables globales asociadas con la sesión, ni destruye la cookie de la sesión. Para destruirla completamente el identificador de sesión también debe ser destruido. Si se usa una cookie para propagar el identificador de sesión, entonces se ha de borrar la cookie de la sesión.
- ***session_name():*** devuelve el nombre de la sesión actual.
- ***session_get_cookie_params():*** obtiene los parámetros de la sesión almacenados en la cookie.

En el apartado siguiente se explicarán el uso de estas funciones con más detenimiento.

3.4.1.2. Aplicación

Una de las páginas dónde se desarrolla la aplicación más importantes es *"trafficlights.php"*. En esta página nos encontraremos con la funcionalidad el juego en la parte del servidor, que junto con el fichero *"functions.js"* en la parte del cliente, realizaran todas las tareas importantes de la aplicación. Lo analizaremos más de cerca para una mayor comprensión de cómo se realizó la aplicación. Para acompañar este análisis se puede encontrar al final de este proyecto un apéndice con los códigos fuente de la aplicación.

"dbfunctions.php"

Antes de empezar vamos a proceder a explicar ahora las funciones de PHP utilizadas para el manejo de la base de datos en el fichero "*dbfunctions.php*" para comprenderlas mejor cuando se utilicen.

```
24 function connect_db() {
25   $link=   mysql_connect(SERVER,USERNAME,PASSWORD);
26   $connection= mysql_select_db(DATABASE,$link);
27   return $link && $connection;
28 } // connect_db
```

Empezaremos con la función de conexión a la base de datos ***mysql_connect()*** que abre una conexión con el servidor MySQL. Es necesario indicar los parámetros de conexión al servidor, como el servidor, el nombre de usuario y la contraseña para una correcta conexión. Después usaremos la función ***mysql_select_db()*** que establece la base de datos activa actual en el servidor asociado con el identificar de enlace especificado que es el obtenido anteriormente por la función anterior. Cada llamada posterior a la función ***mysql_query()*** será ejecutada en la base de datos activa.

```
41 function close_db() {
42   mysql_close();
43 } // close_db
```

La función ***mysql_close()*** cierra la conexión al servidor de MySQL que está asociada con el identificador de enlace especificado. Si no se especifica el identificador del enlace, por defecto utilizará el último enlace abierto.

```php
function obtain_randlist_word(){
    $letter = 'A';

    while($letter!='AA'){
        $res_count= mysql_query("SELECT COUNT(*)
                                 FROM `abecedario`
                                 WHERE letter = '".$letter."'");
        if (!$res_count) {
            echo 'No se pudo ejecutar la consulta: '.mysql_error();
            exit;
        }
        $rowC = mysql_fetch_row($res_count);
        $sql = "SELECT *
                FROM abecedario
                WHERE letter = '".$letter."'
                LIMIT ".mt_rand(0, $rowC[0]-1)." , 1";
        $result = mysql_query($sql);
        if (!$result) {
            echo 'No se pudo ejecutar la consulta: '.mysql_error();
            exit;
        }
        while ($row = mysql_fetch_array($result))
        {
            $list[$letter]['word']=$row['word'];
            $list[$letter]['definition']=$row['definition'];
            $list[$letter]['state']=YELLOW;
        }
        $letter++;
    }
    return $list;
}
```

La función ***obtain_randlist_word()*** se crea para obtener una lista aleatoria de palabras de la base de datos enlazada en ese momento. Empezaremos inicializando la variable *$letter* a la primera letra del alfabeto y crearemos un bucle para que realice las mismas operaciones para las demás letras del abecedario. En el bucle se compará *$letter* con 'AA' porque después de la letra Z la variable toma este valor. Utilizaremos la función ***mysql_query()*** para enviar una única consulta (múltiples consultas a la vez no están soportadas) a la base de datos actualmente activa en el servidor asociado con el identificador de enlace especificado anteriormente. Cómo parámetro de la función utilizaremos lo que llamaremos *"query"* que es una consulta en **SQL** (lenguaje de consulta estructurado, se utiliza para consultar a la base de datos aquellos registros que se desean obtener). En este caso queremos saber el número total de palabras con la letra especificada en ese momento de la tabla abecedario. Si no ocurre ningún error continuaremos con la siguiente función.

La función **m*ysql_fetch_row()*** que devuelve un array número co que corresponde a la fila recuperada y mueve el puntero de datos interno hacia delante o FALSE si no quedan más filas, este valor devuelto lo almacenaremos en la variable *$rowC*. Esta variable la utilizaremos para la siguiente query, en concreto para la función ***mt_rand()*** que se trata de una función mátematica en la librería de PHP utilizada para generar un número aleatorio entre un mínimo y un máximo especificado. En este caso la utilizaremos para generar un número aleatorio entre 0 , el mínimo, y *$rowC - 1*, que será el máximo, en concreto el número total de palabras que hay en la base de datos para esa letra. Se resta 1, porque para indexar las palabras en la base de datos en una matriz se empieza desde 0, sino podría haber errores por intentar acceder a una parte que no existe en la base de datos. Por ejemplo, si tenemos 5 palabras y queremos acceder a la última, esta se encontrará al final de la matriz, es decir en la posición 4.

Volvemos a llamar a la función ***mysql_query()*** esta vez para obtener una palabra aleatoria del total de palabras de esa letra que hay en la base de datos. Una vez obtenido el registro de la base de datos con la palabra y sus datos, si todo ha ido correctamente, procederemos a introducir en la lista la palabra. Para ello utilizaremos la función ***mysql_fetch_array()*** con el resultado obtenido anteriormente para obtener la palabra y la definición y añadir un estado a la palabra.

Ahora explicaremos la estructura del array *$list* por si no queda muy claro en el código. Un array en PHP es realmente un mapa ordenado, es decir un tipo de datos que asocia valores con claves, que puede ser utilizado como una matriz real, una lista (es nuestro caso), una tabla asociativa, y mucho más. Los valores de un array pueden ser otros arrays, árboles y también otros arrays multidimensionales. La estructura es la siguiente:

LISTA →	LETRA →	PALABRA	→VALOR DE PALABRA
		→DEFINICION	→VALOR DE DEFINICION
		→ESTADO	→VALOR DE ESTADO
	→ LETRA →	PALABRA	→VALOR DE PALABRA
		→DEFINICION	→VALOR DE DEFINICION
		→ESTADO	→VALOR DE ESTADO

La lista contendrá un array de letras (desde la A hasta la Z), y en cada letra encontraremos una palabra, una definición y un estado, que serán las claves, y tendrán un valor asociado. Un ejemplo de esto sería por ejemplo:

LISTA → A	→ word	→alarm
	→ definition	→(definición que le corresponde a la palabra)
	→ state	→YELLOW

Tendremos las 3 claves que serán siempre las mismas: word, definition y state, para acceder a sus valores. La última clave sirve para definir el estado de la palabra en cuestión, es decir para decir si es correcta (GREEN), incorrecta (RED), o se ha pasado de la palabra (YELLOW). Utilizaremos el estado YELLOW para indicar un estado inicial que indica un estado nulo y sin importancia. Estos estados son constantes definidas en este mismo archivo:

```
10 DEFINE("RED",0);
11 DEFINE("GREEN",1);
12 DEFINE("YELLOW",2);
```

Se corresponden con números para definir su estado. Al final del bucle inicial tendremos la siguiente línea, *$letter++;* que pasaremos a la siguiente letra para repetir todo el proceso explicado anteriormente. Por último, una vez terminada la lista aleatoria, la devolveremos para guardarla posteriormente en una variable.

Así pues será más fácil la comprensión del siguiente archivo porque sabemos como está estructurada la lista de palabras que utilizaremos prácticamente para todo.

A continuación analizaremos el fichero *"trafficLights.php"* que es importante para procesar los datos que nos envía el usuario desde el cliente web.

“trafficlights.php”

```
3 session_start();
4 include("dbfunctions.php");
5 include("functions.php");
```

En primer lugar, se observa que utilizamos sesiones por la función ***sesión_start()*** explicada anteriormente en la sección de sesiones PHP. Después incluimos dos ficheros php: “*dbfunctions.php*” y “*functions.php*”. En estos dos ficheros se encontrarán tanto funciones para el correcto manejo de la base de datos como funciones generales para un uso más óptimo y reducido de la programación en esta aplicación.

```
 8 $letterlast =  '';
 9 $start = sGetParam("start");
10
11
12 if (isset($start)){
13     $letter = 'A';
14
15     if(!(connect_db()))
16         die ("Error: No se ha podido conectar a la db");
17     else{
18         $list = obtain_randlist_word();
19         close_db();
20     }
21     $_SESSION['list'] = $list; //nueva lista
22     $_SESSION['round'] = 0; //nº de vueltas
23     $_SESSION['end'] = 26; //nº letras hasta finalizar
24 }
```

Aquí están definidas dos variables: *$letterlast* y *$start*. *$letterlast* la utilizamos más adelante para albergar la anterior letra que el juego nos ha enviado y *$start* nos indica el inicio del juego cuando presionamos el botón *start*, que recogeremos con la función auxiliar ***sGetParam()*** usada para conseguir los parámetros enviados por los formularios de la aplicación por HTTP. De tal manera que cuando el parámetro *start* es enviado a “*trafficlights.php*”, lo recogeremos, y en la línea 12 comprobaremos si existe tal parámetro con la función ***isset()***. Esta función está definida en las librerías de PHP y se encarga de determinar si una variable está definida y no es NULL (nula). La única vez que estará definida la variable *$start* será cuando empiece una partida del juego, sino se ejecutarán las siguientes líneas de código.

Empezaremos con la creación de la variable *$letter* inicializándola a la primera letra del alfabeto, la letra A. Esta variable siempre almacenará la letra actual en donde se encuentra el juego en ese momento. Después nos conectaremos a la base de datos con la función ***connect_db()*** siempre y cuando no haya ningún error que en tal caso abortaremos dando parte de ello. Si todo sale bien obtendremos una lista aleatoria de palabras de la base de datos con la función ***obtain_randlist_word()*** y cerraremos la base de datos con ***close_db()*** ya que no utilizaremos la base de datos para nada más. Por último, crearemos las variables de sesiones donde guardaremos todas las variables que necesitamos e iremos utilizando para el buen desarrollo de la aplicación durante la sesión del juego, que son:

- ***list:*** es la lista aleatoria de palabras usada durante el juego. Guardamos la lista para disponer de cada definición y comparar la palabra con la respondida por el usuario para saber si es correcta o no. También guardaremos el estado en que se encuentra cada palabra que inicialmente será YELLOW, como explicamos en el fichero *"dbfunctions.php".*
- ***round:*** *es* el número de vueltas que ha dado el usuario en el juego. Lo guardamos para saber si el usuario ha terminado una vuelta, es decir, que ha alcanzado la letra Z y debe pasar a la siguiente sin responder. También se puede utilizar para futuras mejoras, como por ejemplo, poner un número máximo de vueltas para añadir dificultad al juego.
- ***end:*** es el número de palabras que faltan por contestar para finalizar el juego. Si este número llega a 0, se habrán contestado a todas las palabras y el juego finalizará. Se irá disminuyendo en 1 por cada palabra respondida.

A continuación, si no se ha recibido el parámetro *start* de la aplicación, proseguiremos con el funcionamiento normal del juego:

```
29 else{
30     $list = $_SESSION['list'];
31     $letter = sGetParam("letter");
32     $pass = sGetParam("pass");
33     $wordSpeech = sGetParam("wordSpeech");
34 
35     /* Termina el temporizador */
36     $end = sGetParam("end");
37     if(isset($end)){
38         $_SESSION['end'] = 0;
39     }
```

Siempre comenzaremos por capturar los parámetros que son enviados por el formulario del juego y comprobaremos si están definidos para realizar los cambios necesarios en la lista de palabras y en la aplicación. Esto no quiere decir que todos los parámetros son enviados siempre, por ejemplo, si se pulsa el botón de pasar, el formulario enviará solamente el parámetro pass y letter, dejando los otros sin definir ya que no son necesarios para pasar a la siguiente letra. Para más información del envío de parámetros según el evento que se activa, se encuentra explicado en la sección de JavaScript.

Volviendo al análisis de *"trafficlights.php"*, lo primero que haremos es pasar la lista de palabras de la variable de sesión a una variable local para un mejor uso de la lista, en este caso a la variable *$list*. Después se captura la variable *$letter* que nos indica la letra actual con la que el usuario acaba de realizar una acción*, $pass* que indica si el usuario ha decidido pasar o no*, $wordSpeech* que será la palabra contestada por el usuario y *$end* que si es enviada por el formulario indica que el temporizador de cuenta atrás ha llegado a cero y por tanto cambiaremos la variable de sesión *end* a cero que es la única condición para terminar con el juego y que analizaremos más adelante.

```
46      if(!isset($pass)&&!isset($end)){
47          if($wordSpeech==$list[$letter]['word']){
48              $list[$letter]['state']=GREEN;
49          }else{
50              $list[$letter]['state']=RED;
51          }
52          $_SESSION['end']--;//Contestamos a una letra
53      }
```

En esta captura aparece este fragmento de código para comprobar que si no se ha recibido ni el parámetro *pass* ni el parámetro *end* procederemos a la verificación de la palabra escrita por el usuario en el formulario del juego es la correcta o no, que estará almacenada en la variable $wordSpeech. Para esta verificación se compara con la palabra de la lista de la letra actual en la que ha contestado el usuario. Si coinciden, entonces cambiaremos el estado de la palabra en la lista a GREEN, en caso contrario la palabra será incorrecta y el estado será RED. En cualquier caso, disminuimos en uno la variable de sesión *end* porque hemos contestamos a una palabra, y queda una menos para finalizar el juego.

```
55      $letterlast = $letter; //última letra a la que se ha contestado
56      $letter++; // Se pasa a la siguiente letra
```

A continuación almacenamos la última letra contestada en *$letterlast*, y pasamos a la siguiente letra actual. La primera sentencia la realizamos para que cuando devolvamos la respuesta al cliente, podamos cambiar el semáforo de color y avisar al usuario del resultado de su contestación.

```
59      if($letter=='AA'){
60          $letter = 'A';
61          $_SESSION['round']++;
62      }
```

La segunda acción la utilizamos, como vemos en el fragmento de código para ver si se ha finalizado una vuelta, esto quiere decir que si la última letra a la que el usuario ha contestado es la letra Z, habremos terminado de dar una vuelta en el alfabeto, y en tal caso le daremos el valor 'A' a la variable *$letter* y añadiremos una vuelta a la variable de sesión *round*.

```
64 if($_SESSION['round']!=0&&$_SESSION['end']!=0){
65     while(($list[$letter]['state']!=YELLOW)&&($_SESSION['end']!=0)){
66         $letter++;
67         if($letter=='AA'){
68             $letter = 'A';
69             $_SESSION['round']++;
70         }
71     }
72 }
```

Esta sección la utilizaremos cuando hemos terminado de dar una vuelta y no se ha terminado el juego para buscar aquellas palabras que no se han contestado todavía, es decir, para encontrar las palabras que el usuario ha pasado anteriormente y están en estado YELLOW. De esta forma seleccionaremos la próxima palabra de la lista sin contestar para que el usuario pueda proseguir con el juego.

Si no se ha terminado de dar la vuelta al abecedario, pasaremos a la próxima sección de código.

```php
if($_SESSION['end']==0){
    $countGREEN = 0;
    $countRED = 0;

    for($letterFail='A'; $letterFail!='AA'; $letterFail++){
        if($list[$letterFail]['state']==GREEN)
            $countGREEN++;
        if($list[$letterFail]['state']==RED)
            $countRED++;
    }

    session_unset();
    session_destroy();
    $parametros_cookies = session_get_cookie_params();
    setcookie(session_name(),0,1,$parametros_cookies["path"]);

    die(json_encode(array("countGREEN" => $countGREEN,
                          "countRED" => $countRED,
                          "state" => $list[$letterlast]['state'],
                          "letterlast" =>$letterlast,
                          "end" => "end")));
}
```

Cuando el juego finaliza ya sea porque el usuario ha respondido a todas las palabras o porque se le ha acabado el tiempo, se ejecutará este segmento de código, lo que signica que la variable de sesión *end* ha llegado a cero. Preparamos los resultados para mostrárselos al usuario en el formulario de resultado. Para ello contaremos las palabras que han resultado correctas, y las que son incorrectas, y terminaremos la sesión utilizando las funciones ***sesión_unset()*** y ***sesión_destroy()*** además reiniciaremos las cookies con las funciones ***sesión_get_cookie_params()*** y ***set_cookie()*** para que la próxima vez que un usuario decida empezar la aplicación, no quede guardada ninguna variable de sesión anterior y el juego funcione correctamente. Estas cuatro últimas funciones están explicadas en la sección de sesiones PHP.

Por último devolveremos los resultados al formulario de resultados del juego con la función ***die()*** de la librería PHP, que sirve para finalizar la ejecución del script y el cliente pueda recibir la respuesta del servidor. Dentro de la función añadimos los parámetros del resultado obtenido codificándolos en JSON (Javascript Object Notation) con la función ***json_encode()***. JSON es un formato ligero y simple de intercambio de datos generalmente usado en funciones de AJAX como alternativa de XML. Esta explicado más detalladamente en la sección de AJAX. Los parámetros devueltos son los siguientes:

- ***countGREEN:*** es el número total de palabras acertadas por el usuario.
- ***countRED:*** es el número total de palabras incorrectas contestadas por el usuario.
- ***state:*** es el estado de la última palabra contestada, para que pueda cambiarse el color del semáforo.
- ***letterlast:*** es la última letra de la palabra contestada, sirve para lo mismo que el anterior parámetro.
- ***end:*** indica la finalización del juego para la parte del cliente.

```
105 $_SESSION['list'] = $list;
106 session_write_close();
```

En caso de que el juego no haya terminado, proseguiremos con la continuación del juego. Almacenaremos nuestra lista modificada con el nuevo estado de la palabra actual en la variable de sesión *list* y cerraremos la sesión con la función ***sesión_write_close()***.

```
112 die (json_encode(array("letter" => $letter,
113                        "definition" => $list[$letter]['definition'],
114                        "state" => $list[$letterlast]['state'],
115                        "letterlast" =>$letterlast,
116                        "fin" => $_SESSION['end'])));
```

Finalmente devolveremos el resultado obtenido por el procesamiento de la aplicación al cliente con las mismas funciones utilizadas anteriormente en la finalización del juego, ***die()*** y ***json_encode()***. En este caso devolveremos los siguientes parámetros:

- ***letter:*** la letra actual por la que se encuentra el juego en esos momentos.
- ***definition:*** es la definición de la palabra actual por la que el juego se encuentra en ese momento. La sacamos de la lista junto con la letra actual.
- ***state:*** es el estado de la última palabra contestada, para que pueda cambiarse el color del semáforo.
- ***letterlast:*** es la última letra de la palabra contestada, sirve para lo mismo que el anterior parámetro.
- ***fin:*** es un contador que nos indica cuantas palabras tiene que contestar el usuario para finalizar el juego.

Finalizamos el análisis de los archivos alojados en el servidor que utilizan el lenguaje de programación PHP, y que es una parte importante de procesamiento de la aplicación.

3.4.2. Base de datos con MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de relaciones, pensando en cada relación como si fuese una tabla que está compuesta por *registros* (cada fila de la tabla sería un registro o *tupla*), y *columnas* (también llamadas *campos*).

Se refiere a multihilo a cuando un proceso tiene múltiples hilos de ejecución los cuales realizan actividades distintas, que pueden o no ser cooperativas entre sí. Se le llama multiusuario a la característica de un sistema operativo o programa que permite proveer servicio y procesamiento a múltiples usuarios simultáneamente.

SQL (lenguaje de consulta estructurado) es un lenguaje para el acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

Utilizaremos las sentencias SQL para acceder y obtener las palabras que deseamos de la base de datos como hemos observado en el análisis del fichero *“dbfunctions.php”,* en el apartado de PHP.

3.4.2.1. Creación e importación de la base de datos en el servidor web

En un principio, se utilizó el servidor local *XAMPP*, en el que creamos nuestra base de datos. Para ello utilizamos la herramienta que nos proporciona este servido para la gestión de la base de datos, *PHPMyAdmin*. Con esta herramienta podremos crear, modificar o eliminar bases de datos, tablas y realizar muchas operaciones con ellas, como añadir, modificar o eliminar registros, importar o exportar bases de datos o tablas, etc.

En primer lugar, creamos la base de datos como mostramos en la siguiente captura:

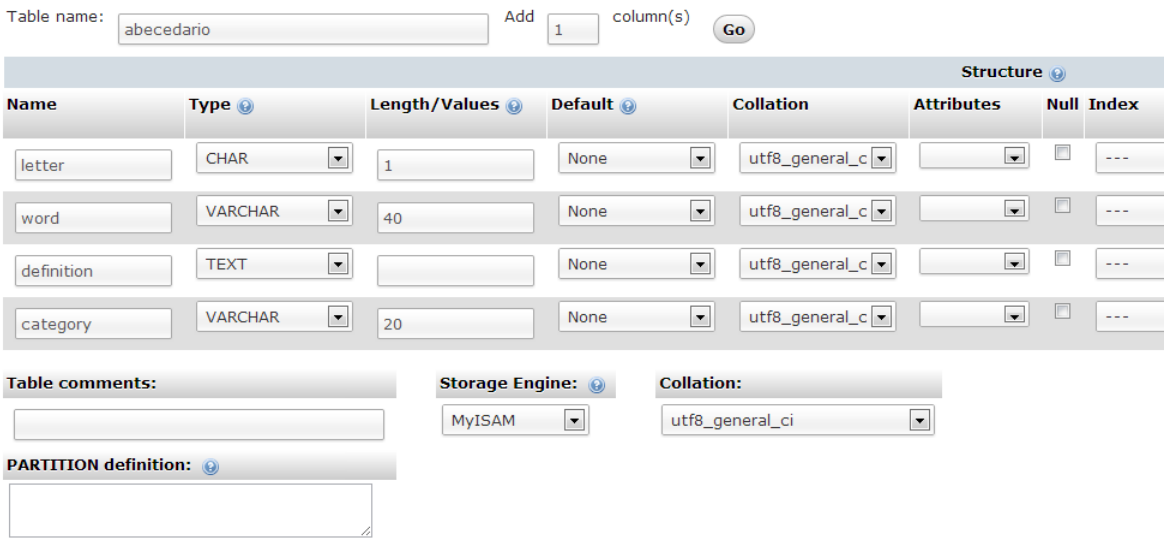


Figura 4.3: Creación de la base de datos en el servidor local

La tabla tendrá el nombre de *abecedario* y contendrá una estructura de cuatro campos: *letter*, *word, definition, category*. Escogeremos un motor de almacenamiento MyISAM, que es la que utilizará el plugin *Adminer* de WordPress. Por último aparecerá la colación, este campo determina el modo en que se realiza la ordenación de la tabla, se seleccionará la más extendida, que será "utf8_general_ci". Pulsaremos el botón *save* y de esta manera habremos creado la tabla que utilizaremos en la aplicación.

El menú de la tabla que nos proporciona *PHPMyAdmin* es el siguiente:

Figura 4.4: Menú de operaciones de una tabla de la base de datos

En *Browse* nos mostrará el contenido de la tabla con los campos que hemos especificado en su estructura al crear la tabla. Como acabamos de crearla estará vacía. *Structure* nos indica la estructura que tiene la tabla para conocer los campos que alberga. *SQL* sirve para realizar consultas de la tabla utilizando este lenguaje. *Search* se utiliza para hacer búsquedas de la tabla según unos operadores, valores y el campo de la estructura que se quiere buscar. *Insert* se encarga de insertar nuevos registros rellenando los campos de la tabla. *Export* se utiliza para exportar nuestra tabla o base de datos a un archivo externo. *Import* se encarga de importar los datos de un archivo a una tabla o base de datos y un formato de archivo concretos. Los demás campos no tienen relevancia en este proyecto por lo que no los explicaremos.

A continuación, insertaremos una por una las palabras escogidas junto con sus definiciones seleccionando la opción *Insert*.

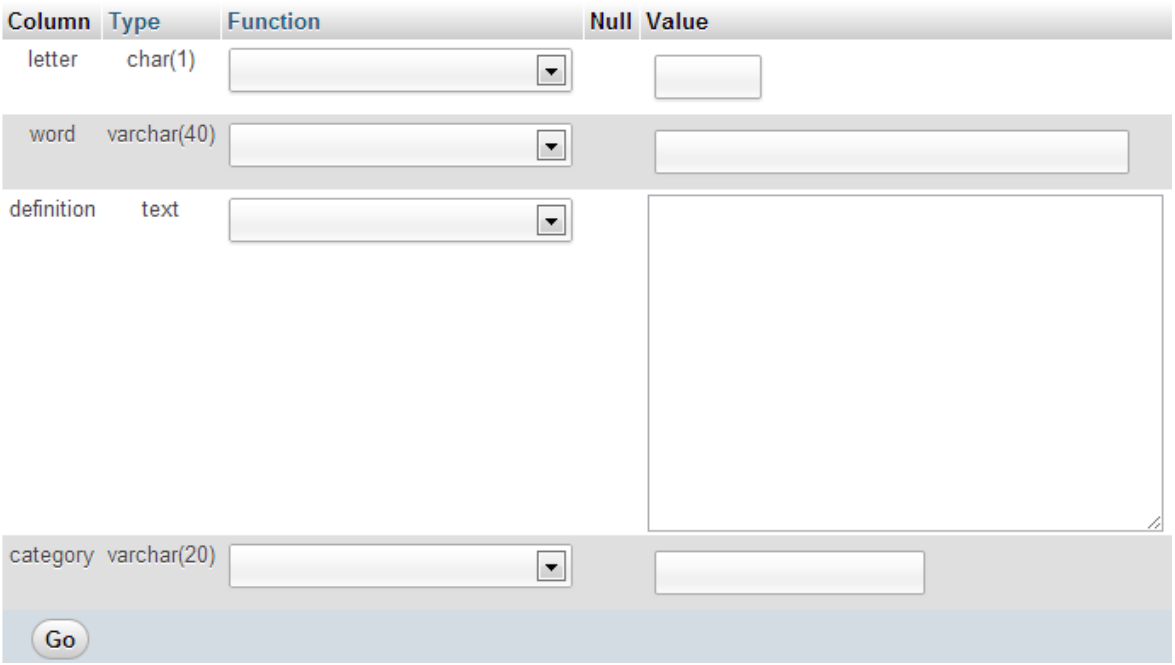
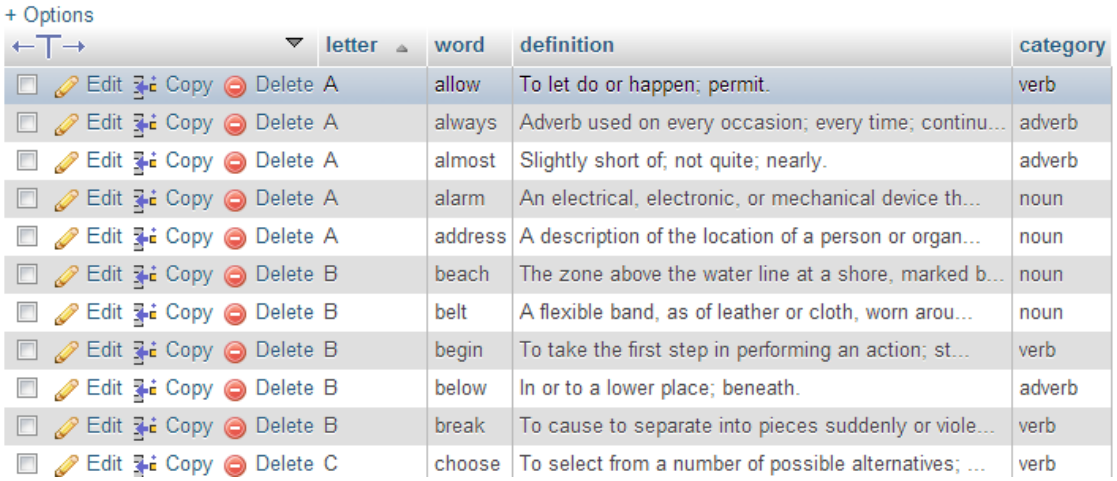


Figura 4.5: Inserción de un registro de la tabla "abecedario"

Rellenaremos el campo *"Value"* de cada campo de la tabla y pulsaremos *"Go"* para introducir el registro en la base de datos. Una vez introducidas todas las palabras, se comprobará en *browse* que todos los campos están insertados correctamente.



+ Options

	letter	word	definition	category
Edit Copy Delete	A	allow	To let do or happen; permit.	verb
Edit Copy Delete	A	always	Adverb used on every occasion; every time; continu...	adverb
Edit Copy Delete	A	almost	Slightly short of; not quite; nearly.	adverb
Edit Copy Delete	A	alarm	An electrical, electronic, or mechanical device th...	noun
Edit Copy Delete	A	address	A description of the location of a person or organ...	noun
Edit Copy Delete	B	beach	The zone above the water line at a shore, marked b...	noun
Edit Copy Delete	B	belt	A flexible band, as of leather or cloth, worn arou...	noun
Edit Copy Delete	B	begin	To take the first step in performing an action; st...	verb
Edit Copy Delete	B	below	In or to a lower place; beneath.	adverb
Edit Copy Delete	B	break	To cause to separate into pieces suddenly or viole...	verb
Edit Copy Delete	C	choose	To select from a number of possible alternatives; ...	verb

Figura 4.6: Visualización del contenido de la tabla "abecedario" en el servidor local

Con la tabla rellena, ya se pueden hacer pruebas en el servidor local *XAMPP*.

Por último, exportaremos nuestra base de datos para poder introducirla en el servidor web ILLLab. Para ello pinchamos en la opción del menú *Export*.

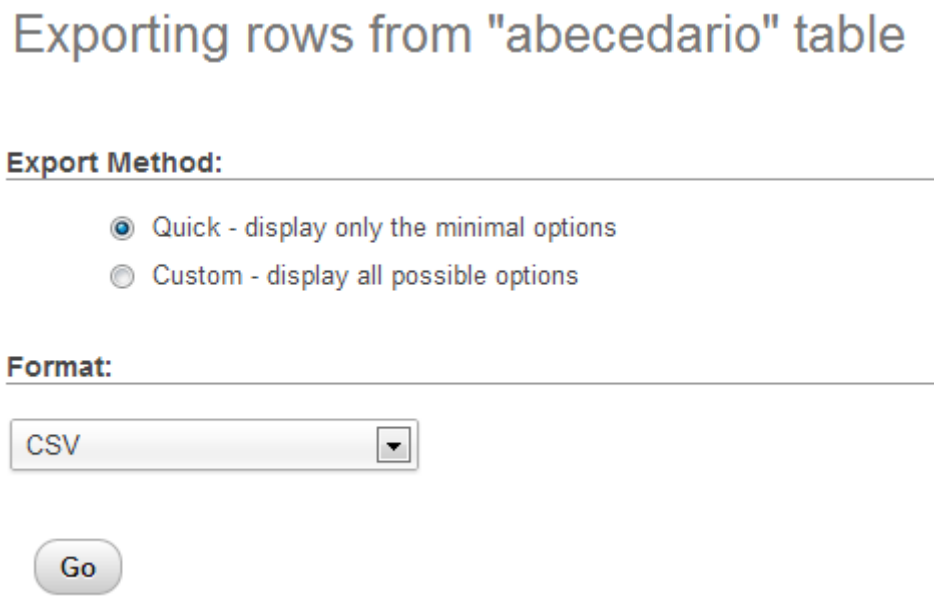


Figura 4.7: Exportación de una tabla de la base de datos en el servidor local

En esta pantalla seleccionaremos la opción Quick, ya que no necesitamos seleccionar ninguna opción avanzada en especial, y pondremos el formato CSV (archivo separado por comas), ya que es el que utiliza *Adminer* en sus tablas de base de datos.

La base de datos que utilizaremos en el servidor web será la que está alojada en ILLLab. Para acceder a ella, utilizaremos el plugin ***Adminer*** de Wordpress. El acceso a la base de datos lo

tendremos que hacer desde el panel de control de WordPress, en concreto en el menú de la izquierda, escogeremos en el menú *Tools* la opción de *Adminer*. Pulsaremos uno de los dos botones que darán acceso a la base de datos de WordPress. Una vez dentro veremos la siguiente interfaz con las tablas de la base de datos:

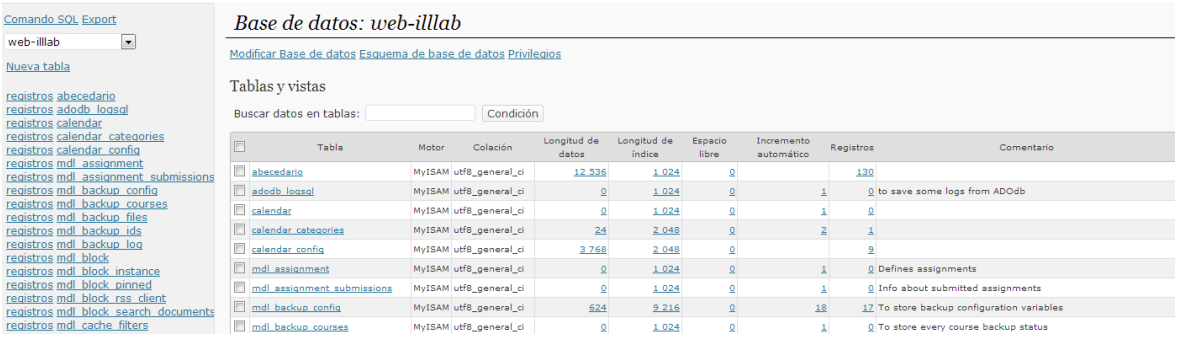


Figura 4.8: Pantalla inicial de la base de datos de WordPress

En esta interfaz se podrá realizar diversas operaciones que se explicarán en la sección correspondiente a este plugin. La que nos interesa es la creación de una nueva tabla donde albergaremos la información que deseamos. Arriba a la izquierda veremos este comando llamado Nueva tabla que pulsaremos para la creación de esta. Aparecerá la siguiente interfaz en la pantalla:

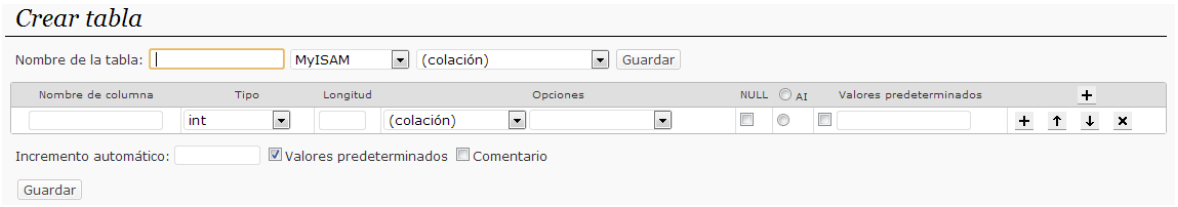


Figura 4.9: Creación de una tabla en la base de datos de WordPress

En esta interfaz aparecen varios campos a rellenar, en primer lugar el nombre de la tabla, que llamaremos “abecedario”. El siguiente campo que aparece será uno de selección y se refiere al mecanismo de almacenamiento que se quiere utilizar, que por defecto aparecerá MyISAM, que es el mecanismo de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales MySQL, y es el que escogeremos. Por último aparecerá la colación, este campo determina el modo en que se realiza la ordenación de la tabla, se seleccionará la más extendida, que será “utf8_general_ci”.

Una vez realizada la tabla pasaremos a la siguiente fila, que aparecerá una tabla para crear la estructura de la tabla de nuestra base de datos. Rellenando todos los campos, la estructura quedará de la siguiente manera:

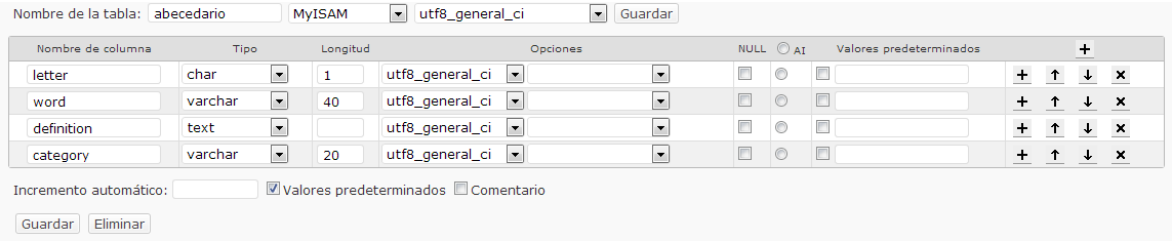


Figura 4.10: Creación de la estructura de la tabla "abecedario" en la base de datos WordPress

Guardamos, y ya tendremos nuestra tabla creada junto con la estructura, el siguiente paso sería añadir las palabras introduciendo nuevos registros, o si ya los tenemos en un archivo, importarlos. Como empezamos con un servidor local, ya teníamos la tabla creada junto con todos los registros, así que en este caso los exportamos para poder importarlos a la base de datos de ILLLab.

Nota: Es importante crear bien la estructura para que la importación sea correcta, sino habrá fallos.

Para importarlo seguiremos los siguientes pasos:

1. En la interfaz de la tabla pulsaremos el enlace de Visualizar contenido:

Figura 4.11: Estructura de la tabla "abecedario" en la base de datos WordPress

2. En la nueva interfaz, bajaremos abajo del todo donde encontraremos la sección de importar, dónde pulsaremos el botón de seleccionar archivo y elegiremos la extensión en la que lo tengamos, en este caso deberemos tenerlo en la extensión *".csv"* para importarlo (se exportará de la anterior base de datos con esta extensión).

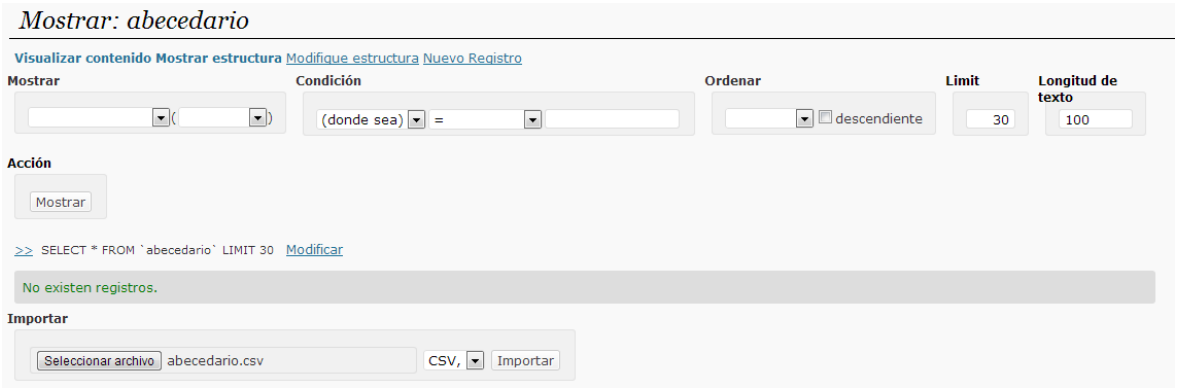


Figura 4.12: Tabla "abecedario" sin registros preparada para la importación de datos

Si todo se ha importado correctamente, aparecerá lo siguiente:

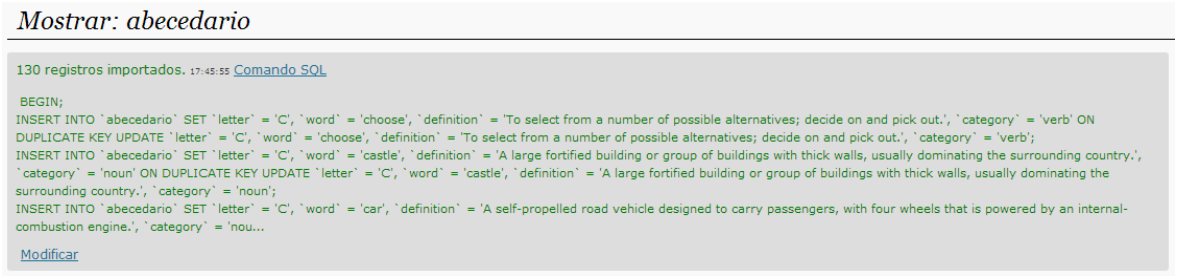


Figura 4.13: Importación de los registros a la tabla "abecedario"

Y ya se podrán visualizar los registros de nuestra tabla perfectamente importados:

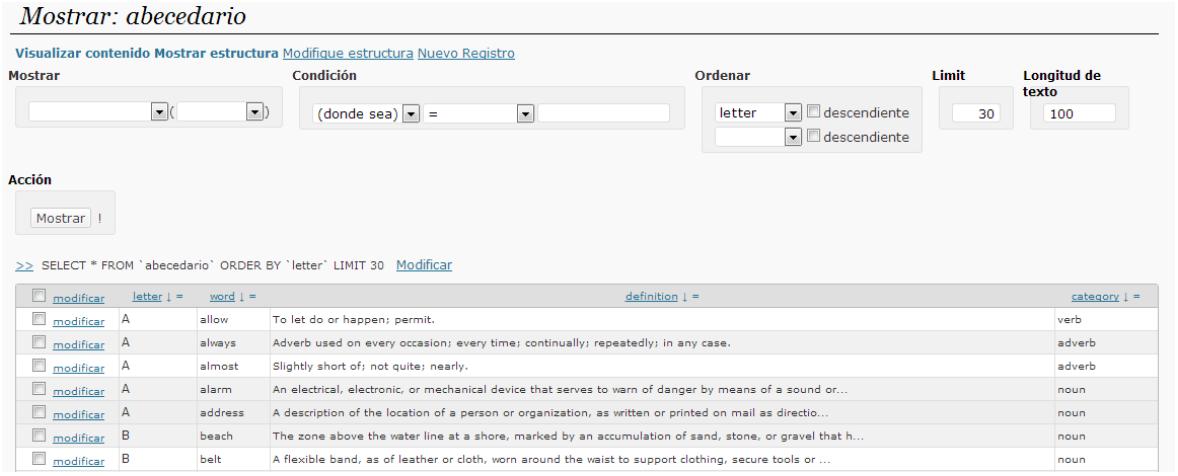


Figura 4.14: Visualización de los registros importados a la tabla "abecederio" en la base de datos de WordPress

Finalmente hay que cambiar los parámetros configurados de la base de datos anterior por la actual en el fichero "*dbfunction.php*" para que la aplicación pueda funcionar correctamente.

En este fichero están definidas unas constantes para la correcta conexión con la base de datos sea más sencilla, de tal manera que con cambiar sólo estas constantes no hará falta nada más.

Las constantes son las siguientes:

```
/***** CONSTANTS *****/

DEFINE("SERVER","server");
DEFINE("USERNAME","user");
DEFINE("PASSWORD","pass");
DEFINE("DATABASE","web-illlab");
```

Nota: Se han cambiado los verdaderos valores para que no haya filtraciones indeseadas. Para obtener estos datos hablar con el tutor de este proyecto para adquirir su aprobación.

La constante SERVER define el servidor donde se encuentra la base de datos, USERNAME es el nombre de usuario de acceso a la base de datos y PASSWORD será la contraseña para ese usuario. Finalmente pondremos el nombre de la base de datos donde alojamos nuestra tabla en DATABASE y guardaremos el archivo, terminando así la importación y correcta configuración de la base de datos para el uso correcto de la aplicación.

3.5. Servidor Local Xampp

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas.

Oficialmente, los diseñadores de XAMPP sólo pretendían su uso como una herramienta de desarrollo, para permitir a los diseñadores de sitios webs y programadores testear su trabajo en sus propios ordenadores sin ningún acceso a Internet, es decir, localmente. En la práctica, sin embargo, XAMPP es utilizado actualmente como servidor de sitios Web, ya que, con algunas modificaciones, es generalmente lo suficientemente seguro para serlo.

Para la realización de este proyecto utilizamos este servidor que nos provee de las herramientas necesarias para probar nuestra aplicación de forma local. En nuestro caso sólo utilizamos el servidor web Apache para interpretar PHP y MySQL para gestionar la base de datos. En concreto utilizamos una versión portable (no es necesaria una instalación en el sistema operativo) con una interfaz para controlar y configurar los servidores:

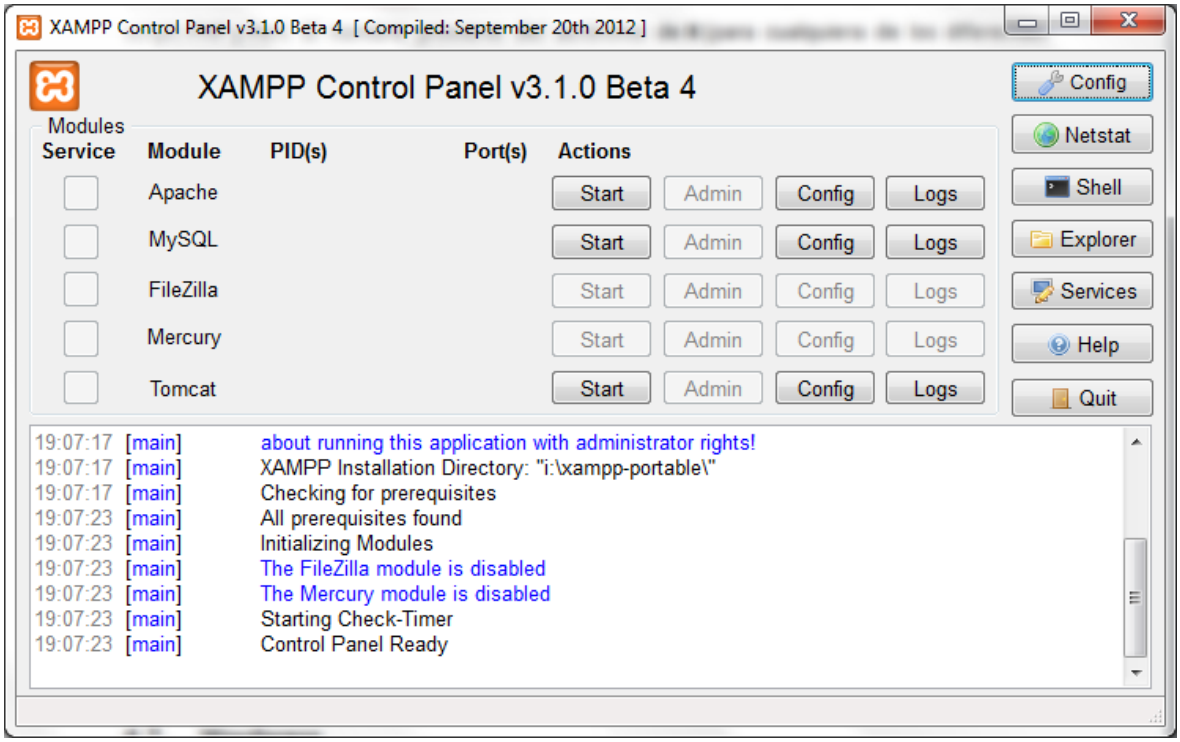


Figura 4.15: Panel de Control del servidor local XAMPP

Al poner a funcionar los servidores Apache y MySQL, podemos probar nuestra aplicación localmente sin conexión a internet. En la siguiente captura vemos a los servidores activos:

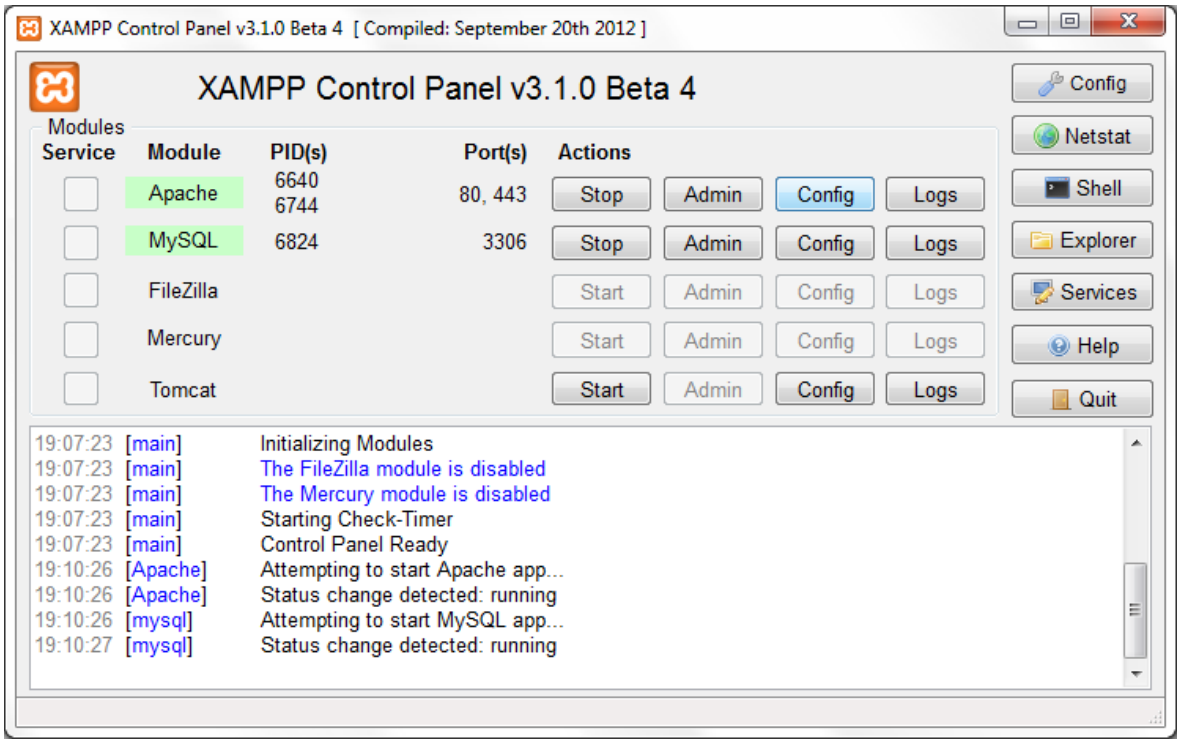
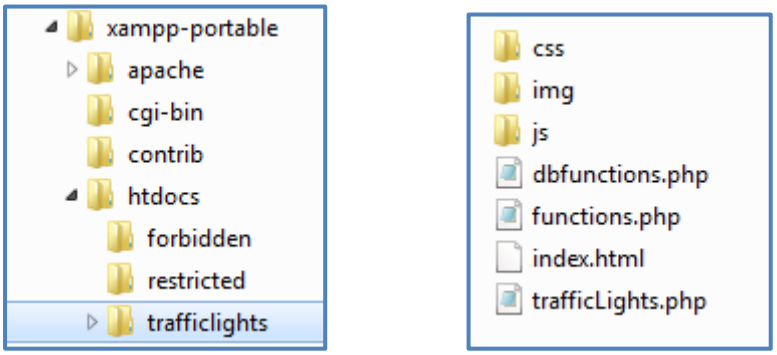


Figura 4.16: Activación de los servidores locales en el panel de control XAMPP

Lo único que se ha de saber para poder probar nuestra aplicación es el directorio donde deberemos incluir nuestro proyecto. Este directorio es *xampp-portable\htdocs*, dentro de él meteremos nuestra carpeta en la cual contendrá nuestros archivos html, css, php y js, además de otros ficheros como imágenes que utilicemos en la página web, etc.



Para acceder a la página web de nuestra aplicación, deberemos abrir el navegador e introducir la siguiente dirección: http://localhost/trafficlights/index.html. Tras pinchar en el link accederíamos al inicio de nuestra aplicación como si estuviera en un servidor web.

Cualquier carpeta que introduzcamos, con documentos web, en htdocs, podrá ser accedida desde el navegador de la misma manera que nuestra aplicación, posibilitando al desarrollador web hacer todo tipo de pruebas.

Este servidor dispone de un panel de control que se puede acceder poniendo la siguiente dirección: http://localhost/xampp/.

Figura 4.17: Página de gestión del servidor local XAMPP

En este panel de control podemos observar que incluye diferentes herramientas para configurar los diferentes servidores y aplicaciones para su manejo. Una de ellas de las cuales utilizamos en el manejo de base de datos MySQL en este servidor local es *PHPMyAdmin*.

PHPMyAdmin es un gestor de base de datos en el que crearemos nuestra base de datos, en concreto la tabla *abecedario*, que es la que utilizaremos en este proyecto. En la sección 4.4.2.1 de Base de Datos se explica la creación de la tabla utilizada en la aplicación, por lo que no entraremos en más detalles en esta sección sobre esta herramienta.

3.6. Servidor Web ILLLab

El servidor Web de ILLLab está alojado en la EUITT de la UPM. Estos servidores web son muy seguros debido a que están muy protegidos y su acceso es muy restrictivo, pudiendo solamente acceder el personal de la UPM. Para poder acceder a este servidor el administrador de la página web de ILLLab contactó con el servicio técnico de la EUITT para poder tener acceso a él, proporcionándonos el usuario y la contraseña.

Una vez obtenidas, se pudo subir la aplicación TrafficLights al servidor vía FTP. En el manual de usuario se puede ver un ejemplo de cómo usamos FTP para modificar un archivo. Además para acceder a la base de datos de ILLLab, utilizamos un plugin de Wordpress llamado Adminer, explicados en el siguiente apartado.

3.7. WordPress

WordPress es un CMS o sistema de gestión de contenidos en la que puedes escribir, modificar artículos y crear una página web o un blog. Es popularmente usado por su facilidad y sus características como gestor de contenidos, además de la enorme comunidad de desarrolladores y diseñadores web que desarrollan plugins y temas para la comunidad de WordPress. Es bastante sencillo de utilizar, por lo que usuarios inexpertos en conocimientos sobre desarrollo web pueden empezar a crear y gestionar páginas web siguiendo paso a paso tutoriales creados por la comunidad de WordPress.

En principio, Wordpress es un sistema de publicación web basado en entradas ordenadas por fechas, entre otras muchas posibilidades además de páginas estáticas. La estructura y diseño visual del sitio depende de un sistema de plantillas, independiente del contenido en sí. Esto se debe a que el diseño está separado del contenido de la página, haciendo posible un cambio rápido de diseño sin tener que modificar el contenido. Para conseguir esto, WordPress divide la información que conforma la web en **post** y **páginas**, de manera que presenta una forma de uso similar a la de un blog. En cuanto al diseño, dependerá del tema seleccionado en ese momento por el creador.

Un **post** o entrada es el sistema básico para introducir cualquier tipo de contenido en WordPress. Se destinan a alojar y dar visibilidad a las diferentes informaciones contenidas en la web. A diferencia de las páginas, los post normalmente cuentan con uno o varios campos destinados a que los usuarios escriban sus comentarios. Además, a no ser que el administrador indique lo contrario, estos se muestran en orden cronológico inverso en la página principal de la web.

Una **página**, en WordPress, está destinada a alojar un contenido estático, distinto de los post. Por ejemplo, pueden utilizarse para mostrar distintos post, filtrando estos por categorías, facilitando al usuario la lectura y navegación entre de las diferentes informaciones contenidas a través del conjunto de post que conforman la web. Asimismo, sirven para almacenar datos destinados a tener poca variación a lo largo del tiempo de vida de la web, como la información de contacto, datos del creador de la página, etc.

A diferencia de los post, su contenido suele estar en forma de menús que aparecen en la parte superior de la página, de manera que es fácilmente accesible desde cualquier punto de la web. Otra diferencia importante entre post y páginas es que en las páginas además de no permitir comentarios, no se muestra la fecha de publicación o el usuario. Su contenido tampoco aparece en el listado de post de la página principal de la web.

Cómo nuestra aplicación está alojada en una página aparte de WordPress, no indagaremos más en la explicación de lo que se puede o no hacer en WordPress, ya que para eso existe una gran comunidad de la que se puede extraer cualquier información que se necesite. Ahora que conocemos un poco más WordPress, vamos a ver como se instala el plugin *Adminer* que nos dará acceso a la base de datos de la aplicación, pero antes aclararemos que es un plugin y para qué sirve.

Un plugin es un complemento que se añade a una aplicación para aportarle una función nueva y generalmente muy específica. Un plugin permite que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones. Todos y cada uno de los plugins disponibles para su descarga e instalación en el panel de control de WordPress han sido desarrollados con ánimo de resultar fáciles de manejar para un usuario con escaso conocimiento en desarrollo de aplicaciones web. En nuestro caso la aplicación principal es WordPress que por sí sola no tiene la función de gestionar su base de datos, y nuestro complemento es Adminer, que añade la capacidad de gestionar la base de datos que comprende WordPress de manera sencilla.

Para la instalación como la desinstalación de los diferentes plugins que pueden incluirse en WordPress puede ser fácilmente realizada desde el panel de control. Basta con entrar en el menú de *plugins*, disponible entre la lista de menús del administrador localizada en la parte lateral izquierda del panel de control.

Al pinchar en esta opción, accederemos a un listado donde se nos muestran los diferentes plugins instalados. Para cada uno de ellos, tenemos la opción tanto de desactivarlo como de desinstalarlo con sólo pulsar un enlace. Asimismo, entre la información proporcionada sobre cada uno de los plugins instalados, se encuentra la dirección web de la página del plugin, a la que podemos acceder en el caso de necesitar información adicional sobre un plugin en concreto o su funcionamiento.

Al pulsar la opción de *Plugins* del menú del administrador en el panel de control, obtendremos un listado con apariencia similar al siguiente:

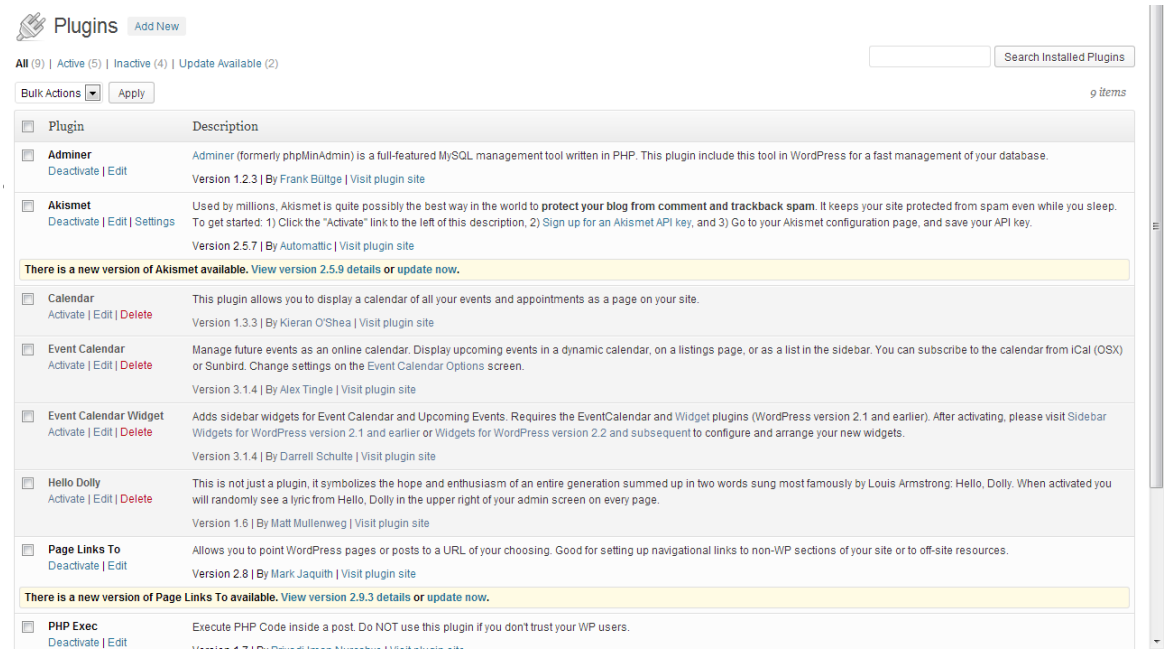


Figura 4.18: Página de plugins del escritorio de WordPress

Además de las funciones mostradas anteriormente, a la derecha del título *Plugins* aparece la opción "*Add New*", con la que se puede instalar cualquier plugin que necesitemos de forma simple, como describiremos a continuación.

Al pinchar en el botón "*Add new*", aparecerán una serie de opciones, mostradas en la siguiente figura:

Figura 4.19: Página de búsqueda de plugins

Como podemos ver, tenemos la posibilidad de instalar el plugin deseado de diferentes formas. A continuación, vamos a describir brevemente las más importantes de ellas.

El primero de los enlaces, *Upload*, nos permite instalar un plugin si ya tenemos los archivos que lo forman descargados previamente. Estos archivos deben estar incluidos en un archivo con extensión *.zip* para que WordPress pueda instalar el plugin correctamente.

El resto de los enlaces que aparecen, *Featured*, *Popular*, *Newest* y *Favorites* nos lleva a un listado de los plugins que cumplen las condiciones descritas en el enlace correspondiente.

A continuación, bajo el epígrafe *Search*, tenemos la opción de realizar una búsqueda en el repositorio oficial de plugins de WordPress, de manera que podamos encontrar e instalar un plugin que aporte a nuestra web la funcionalidad que estamos buscando. Podemos realizar esta búsqueda introduciendo una o más palabras clave. Las búsquedas pueden realizarse atendiendo al Término, o nombre del plugin, autor del mismo o etiqueta o tag bajo la que se haya categorizado.

Por último, en la parte inferior, bajo el epígrafe *Popular Tags*, podemos encontrar una nube de *tags* o etiquetas, formada por las etiquetas bajo las que se encuentran clasificados los plugins más frecuentemente instalados por el resto de usuarios de WordPress.

Utilizando la opción de *Search*, buscamos el plugin que deseamos instalar, *Adminer*, el cuál aparecerá el primero de la lista como se muestra en la siguiente imagen:

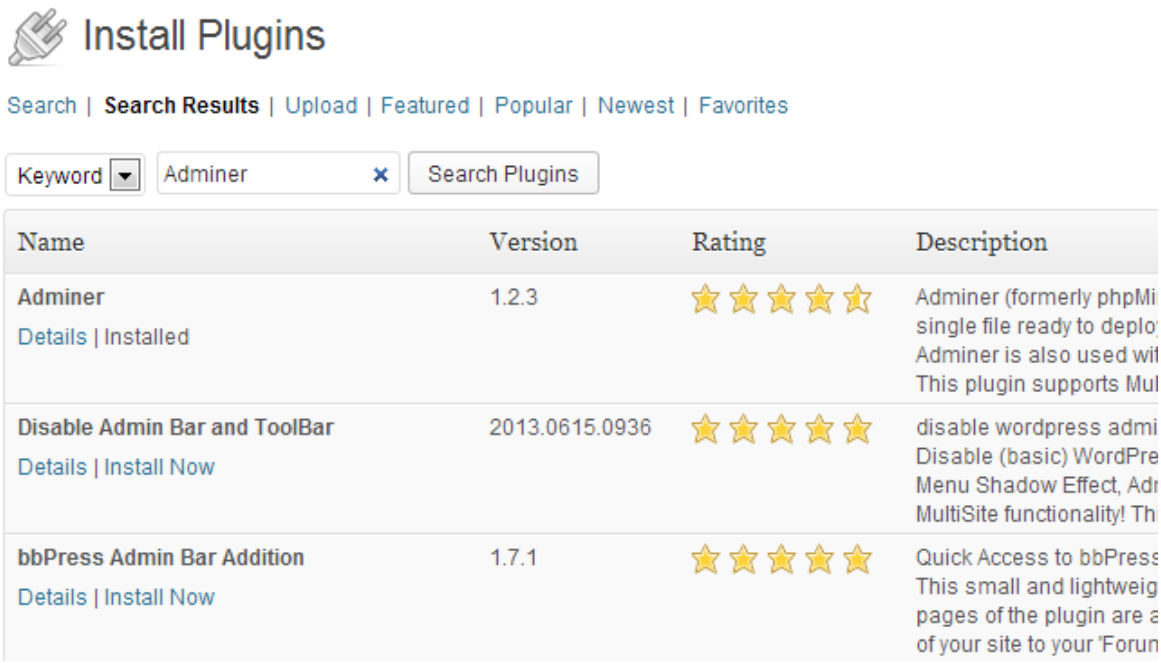


Figura 4.20: Instalación de plugins

En nuestro WordPress ya aparece instalado, pero para instalarlo se deberá pulsar la opción *"Install Now"* que aparece debajo del nombre del plugin, como puede verse en los demás plugins.

Los diferentes plugins que pueden instalarse en WordPress resultan fácilmente manejables, de manera que incluirlos en el desarrollo de nuestro trabajo no supone un incremento sustancial de la dificultad de manejo del Panel de Control. Sin embargo, el alto valor que tiene para nuestro desarrollo la funcionalidad añadida que proporciona esto, justifica su uso en la implementación de nuestra herramienta de consulta.

A continuación, vamos a describir el plugin Adminer utilizado en la elaboración de este proyecto, explicando su funcionalidad concreta y justificando su uso para alcanzar nuestro propósito final.

3.7.1. Adminer

Adminer es un gestor de bases de datos que ofrece la administración completa de bases de datos MySQL y que dispone de un plugin para WordPress, lo que significa que tendremos acceso a nuestra base de datos desde el panel de control de WordPress.

Este plugin nos permite acceder a la base de datos de WordPress, de manera que se pueden crear y modificar las tablas, así como los valores de los diferentes registros. En nuestro caso, además, nos ofrece una ventaja adicional: permite tanto exportar como importar el contenido de una tabla, a través de un archivo CSV (archivo separado por comas), que puede ser modificado de manera sencilla tanto por editores de texto como por aplicaciones de hojas de cálculo.

Como inicialmente para la realización de nuestro proyecto se realizó en un servidor local, se utilizó la herramienta *PHPMyAdmin*, que viene por defecto en el panel de control de *XAMPP*, para la creación de nuestra base de datos. Con *PHPMyAdmin*, también tenemos la opción de importar o exportar una base de datos en diferentes formatos, uno de ellos es CSV, que es justamente el que admite *Adminer*. Una vez que disponemos de nuestra base de datos exportada con la extensión *.csv*, procederemos a importarla en la base de datos de WordPress con el plugin *Adminer*.

En este apartado vamos a explicar la interfaz general de *Adminer* aunque en otras secciones del proyecto ya se han incluido muchas de las funcionalidades que tiene y que usamos.

Al instalar este plugin, aparece en el menú *Tools* del panel de control de WordPress la opción *Adminer*. Una vez que pinchamos en él aparece la siguiente pantalla:

Figura 4.21: Pantalla de inicio del plugin Adminer

Con esto, podemos elegir visualizar la base de datos en la misma ventana, o en una ventana independiente. Una vez que elegimos la opción deseada, se nos muestran las diferentes tablas que componen la base de datos de nuestro WordPress de una forma similar a la siguiente imagen:

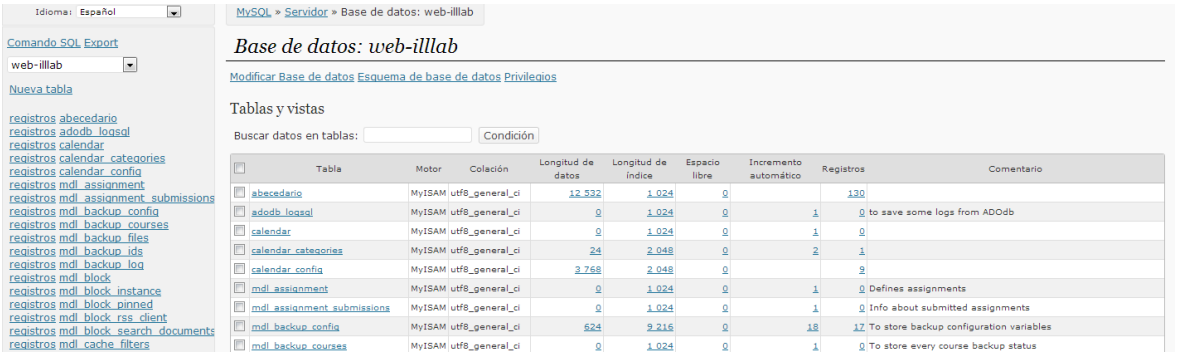


Figura 4.22: Pantalla de la base de datos de WordPress

Como vemos, por cada tabla se nos muestra una serie de información básica, como el motor de búsqueda, la codificación utilizada, el número de registros, etc.

Para visualizar o modificar el contenido de alguna de las tablas, sólo tenemos que pinchar sobre su nombre, tanto en el listado de la izquierda como en el central. Esto nos conduce a una información más personalizada de la tabla elegida, con aspecto similar al de la siguiente imagen:

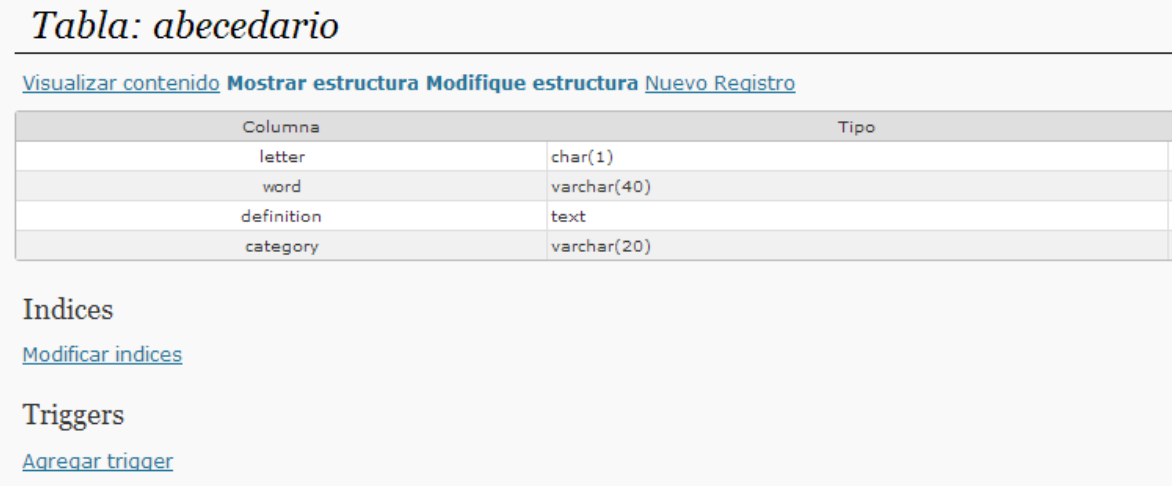


Figura 4.23: Estructura de la tabla "abecedario"

Como se muestra en la imagen, tenemos acceso a diferentes acciones posibles a realizar sobre la tabla seleccionada, como ver su estructura, el nombre de los campos, su tipo, y los índices agregados. Además, desde aquí podemos ver el contenido de los diferentes registros, modificar la estructura de la misma tabla, o añadir índices o registros nuevos.

Para añadir nuevos registros, o modificar los ya existentes, debemos pulsar en *Visualizar Contenido*. Al hacer esto, obtendremos una vista similar a esta:

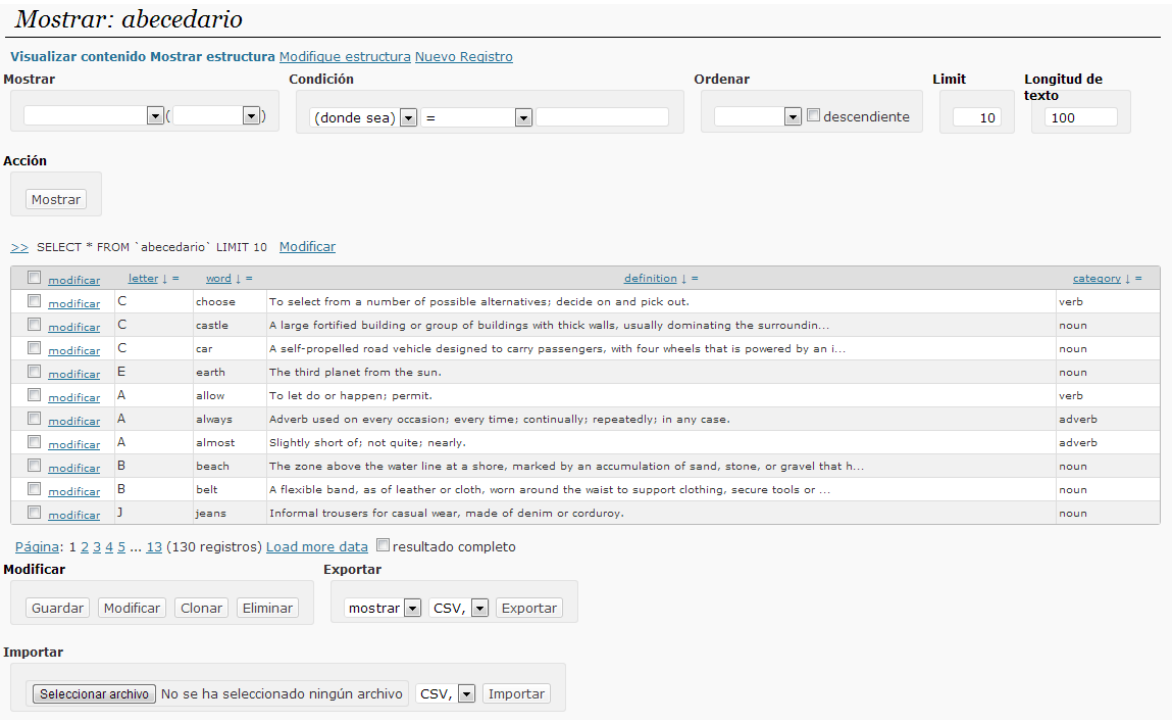


Figura 4.24: Visualización del contenido de la tabla "abecedario"

Como se puede observar en la imagen, además de las opciones de la vista anterior, se nos muestran los primeros registros de la tabla. En los apartados *Mostrar*, *Condición* y *Ordenar* podemos elegir entre una serie de funciones SQL y condiciones de búsqueda, de manera que se nos muestren todos los registros coincidentes. Estas opciones son muy útiles a la hora de diseñar diferentes condiciones de búsqueda, ya que nos permiten probarlas directamente en la base de datos sin necesidad de codificar y ejecutar ningún programa adicional. Asimismo, existe la posibilidad de introducir directamente la función y la condición o condiciones de consulta manualmente, utilizando para ello la opción *Comando SQL* que aparece en la parte superior del menú de la izquierda, tal y como se muestra en la siguiente imagen:

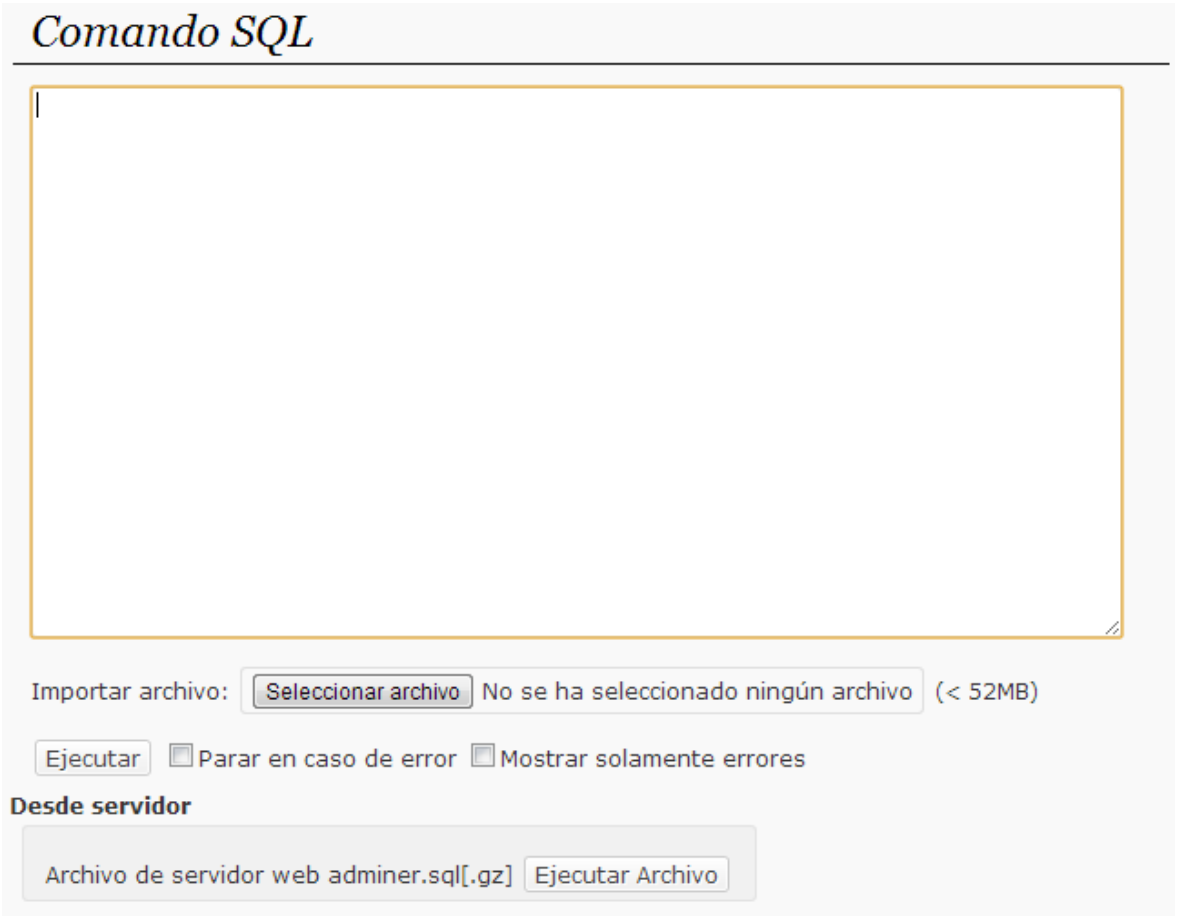


Figura 4.25: Pantalla para realizar consultas a la tabla mediante comandos SQL

Con escribir la expresión deseada y pulsar el botón *Ejecutar* se realiza la acción descrita en la expresión sobre la base de datos, mostrándose el resultado de la misma.

En la parte inferior de la página de visualización del contenido de la tabla tenemos la opción de avanzar entre los diferentes registros de la tabla, eliminar los que seleccionemos, o importar y exportar su contenido, tal y como podemos apreciar en la siguiente imagen:

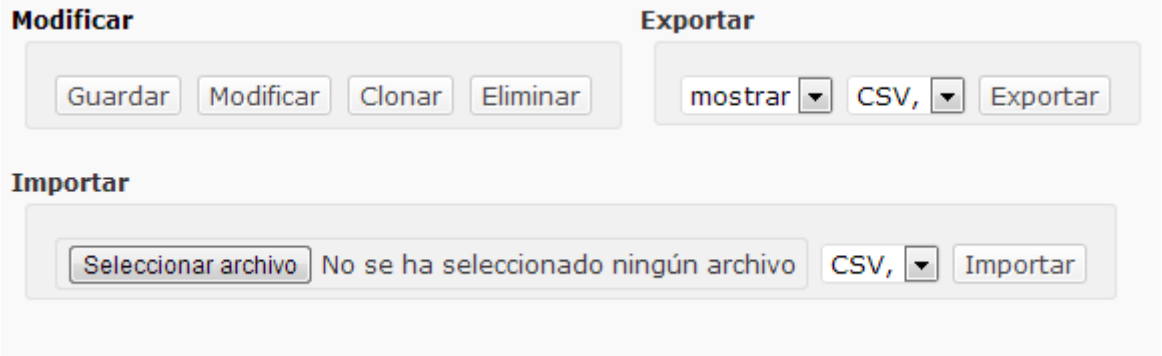


Figura 4.26: Menús para la modificación, exportación e importación de tablas de la base de datos

En el menú de *Modificar* vemos que tendremos las opciones de *Guardar*, *Modificar, Clonar,* y *Eliminar.* Con este menú podremos modificar los distintos campos de la tabla así como la tabla en sí según las diferentes opciones que hemos visto que ofrece.

Exportar se encargará de exportar nuestra tabla a un archivo con una extensión que seleccionemos, aunque por defecto esta elegido "*.csv".*

Por último encontraremos *Importar*. Al pulsar en el botón *Examinar* se nos abre un cuadro de diálogo desde el cual podemos elegir el archivo a importar. Una vez hecho esto, al hacer clic sobre el botón *Importar* los datos que contienen en archivo son introducidos en la tabla seleccionada.

Es importante señalar que para que esta herramienta de importación funcione correctamente el archivo CSV debe corresponderse con una tabla que tenga el mismo número de columnas, nombradas idénticamente, que la tabla destino en nuestra base de datos. En caso de no tener ambas el mismo número de columnas, la herramienta de importación no será capaz de identificar correctamente los diferentes campos a añadir a la tabla, por lo que mostrará un mensaje de error. Si, aun teniendo el mismo número de columnas, estas tienen un nombre diferente, la fila correspondiente a los nombres de las columnas en la tabla de origen será interpretada por la herramienta de importación como un registro más, por lo que se añadirá su contenido a la tabla.

Para garantizar el correcto funcionamiento de la herramienta de importación, dentro del archivo CSV todos los campos a importar deben ir delimitados por el carácter "," tanto al principio como al final del campo. En caso contrario no se producirá la importación, mostrándose un mensaje de error. Por último, comentar que es posible importar tanto archivos separados por el carácter "," como por el carácter ";". En nuestro caso, y dado que algunos de los campos a importar contaban con el carácter "," en su contenido, se decidió utilizar el carácter ";" para prevenir posibles errores en la importación de datos.

4. CAPÍTULO 4: MANUAL DE USUARIO

En las páginas siguientes, explicaremos a través de un documento en formato de manual cómo funciona la aplicación; indicaremos las reglas del juego desarrollado, mostraremos un ejemplo de su uso, explicaremos como añadir nuevas palabras a la base de datos de la aplicación y cómo cambiar el tiempo estimado de finalización del juego.

4.1. Pantalla inicial de la aplicación

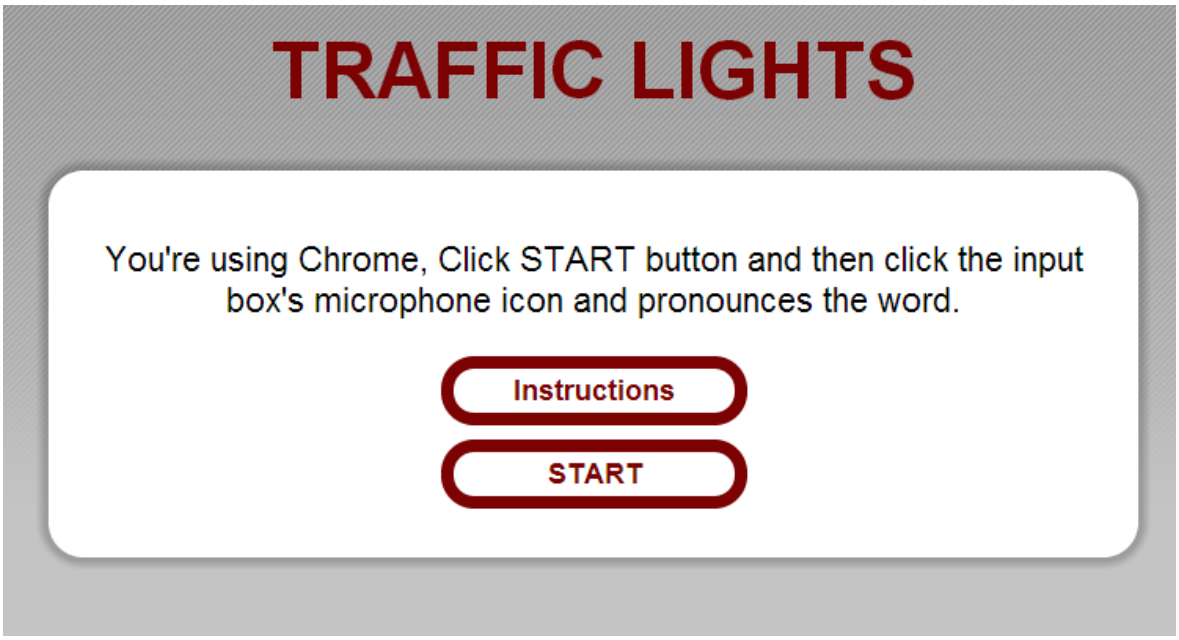


Figura 5.1: Pantalla de inicio de *Traffic Lights*

La pantalla de inicio de la aplicación consta del título de la aplicación, de un pequeño párrafo y dos botones. El texto escrito nos indica si estamos usando Google Chrome con su versión 11 o superior para poder hacer uso de la herramienta de reconocimiento de voz que nos proporciona. Podremos utilizar la opción de reconocimiento de voz mediante un pequeño micrófono que aparecerá en una caja de texto que se mostrará una vez iniciado la aplicación. Después nos encontramos con los botones de “instructions”, que mostrará las instrucciones del juego como veremos en el siguiente apartado y “start” que iniciará la aplicación.

4.2. Instrucciones y reglas de la aplicación

El botón de instrucciones mostrará las instrucciones de la aplicación para que el usuario conozca cómo funciona el juego y en qué consisten sus reglas. Ahora vamos a explicarlas detalladamente.

Traffic Lights es una aplicación basada en un juego de alfabeto, es decir, que el usuario deberá adivinar palabra por palabra desde el principio hasta el final del alfabeto, desde la A hasta la Z, de modo que cada vez que el usuario conteste a una palabra pasará a la siguiente letra del abecedario hasta finalizar.

Para adivinar la palabra se proporciona la definición de ésta y el usuario tendrá que averiguarla. Asimismo, para un mayor desafío para el usuario, se ha añadido un temporizador con cuenta atrás que comienza con unos pocos minutos.

Además, con objeto de amenizar el juego se han añadido las siguientes reglas:

- Si el usuario adivina la palabra, una imagen de un semáforo debajo de la letra en la que se encuentra en ese momento el juego, se pondrá en verde y se añadirán 5 segundos a su temporizador.
- Si el usuario falla al intentar adivinar la palabra, la imagen del semáforo se cambiará a rojo y se restarán 5 segundos de su temporizador.

Además se podrá pasar de letra si el usuario no sabe la palabra de tal manera que podrá contestar en la siguiente ronda.

El juego terminará cuando se contesten a todas las palabras (sean o no correctas) o el temporizador llegue a cero y entonces aparecerá la pantalla con los resultados que el usuario ha obtenido.

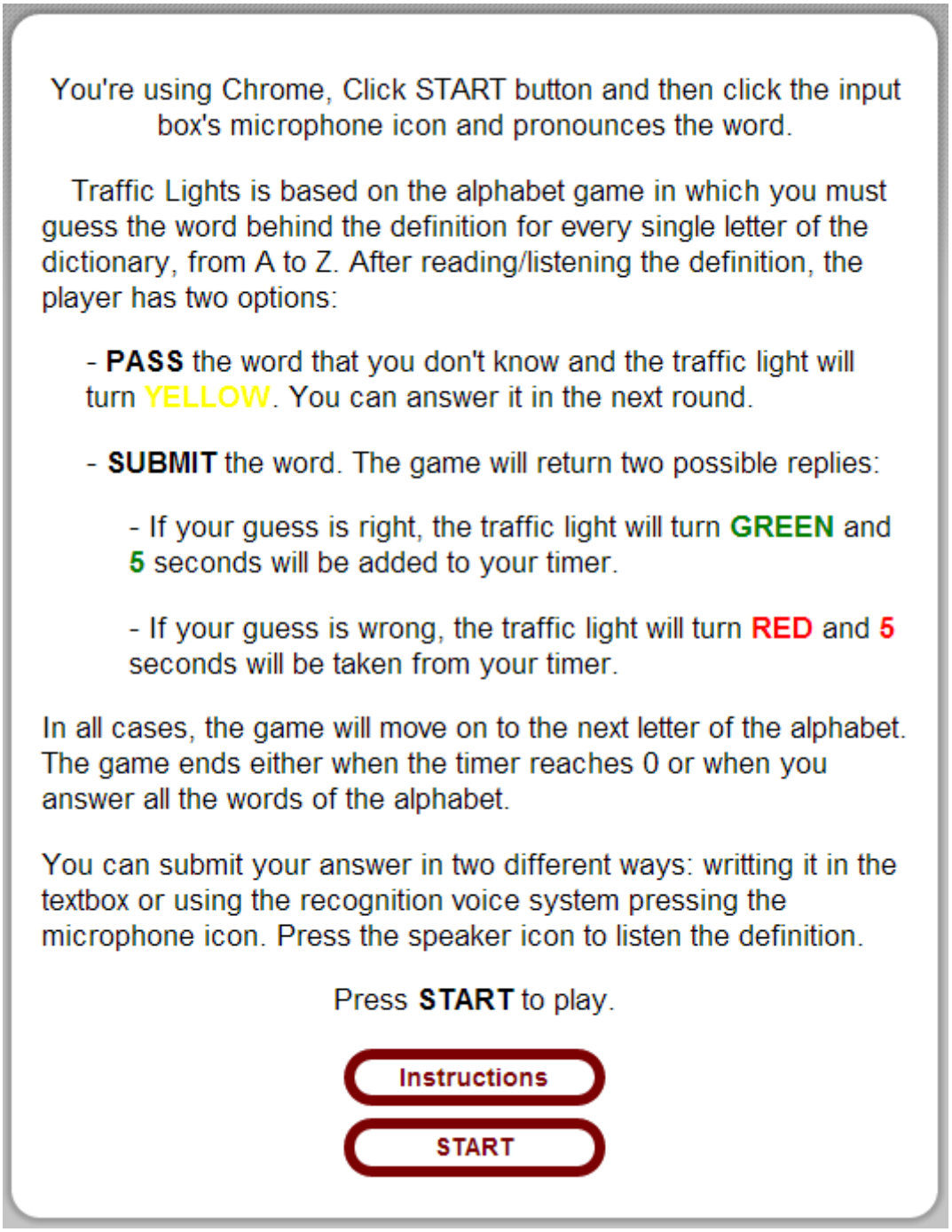


Figura 5.2: Instrucciones de *Traffic Lights*

A continuación mostraremos un ejemplo de uso de la aplicación.

4.3. Ejemplo de uso

4.3.1. Empezar

Para empezar deberemos pulsar el botón de START. Una vez pulsado, el juego cambiará a la pantalla principal de la aplicación donde se desarrollará el juego.

Figura 5.3: Pulsar START para empezar una partida

4.3.2. Letra a letra

Una vez empezado el juego aparecerá la pantalla siguiente:

Figura 5.4: Pantalla de juego de *Traffic Lights*

En esta pantalla nos encontramos con los siguientes elementos:

- Una **fila de semáforos** apagados; cada uno de ellos se relaciona con cada una de las letras del abecedario, desde la A hasta la Z, de esta manera el usuario sabrá en cada momento cuantos aciertos, fallos y palabras sin contestar ha tenido.
- El **temporizador,** que indicará siempre el tiempo del que dispone el usuario para finalizar el juego.
- El **bloque principal** del juego; aquí se puede observar que da a conocer al usuario la *letra* en la que se encuentra en ese momento así como su *definición* correspondiente. Además tendrá la *caja de texto* donde el usuario escribirá la palabra correspondiente. La caja de texto introduce la opción del micrófono. Para hablar por el micrófono, se deberá pulsar con el ratón el dibujo del *micrófono* a la derecha de la caja de texto, en que aparecerá una barra que indica el nivel de audio que recibe por el micrófono.

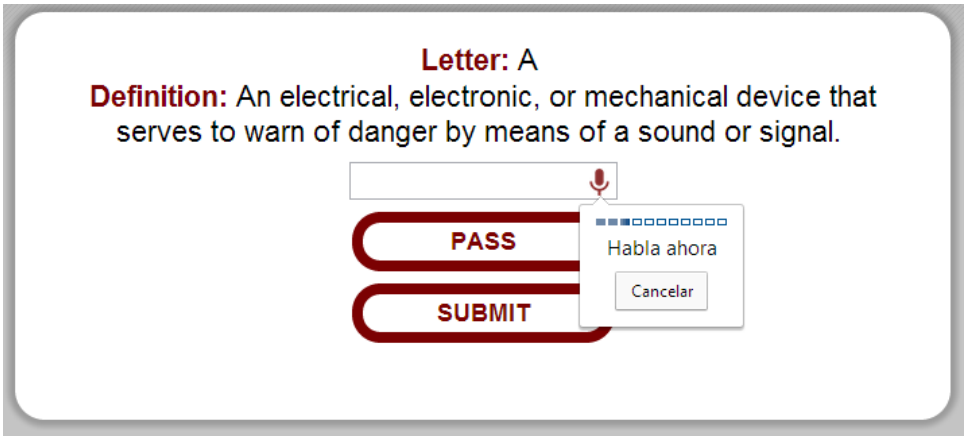


Figura 5.5: Funcionamiento del reconocimiento de la voz

Al pasar unos segundos si el reconocedor de voz no detecta ninguna voz o no reconoce la voz que ha recibido por la entrada de audio, nos mostrará un mensaje como el siguiente:

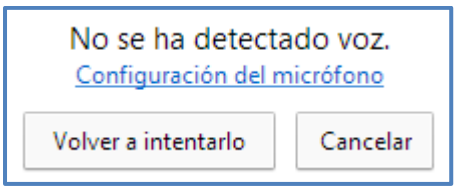


Figura 5.6: Mensaje de aviso del reconocedor de voz

Se podrá volver a intentar o cancelar si no se desea volver a utilizar esta opción.

Por último se encuentran los botones “PASS” y “SUBMIT”. El botón “PASS” sirve para pasar a la siguiente letra si se desconoce la palabra de la letra actual, y el botón “SUBMIT” sirve para enviar la palabra a la aplicación y que está lo procese devolviendo el resultado (si es correcto o no) y pasando a la siguiente letra, no antes de que se encienda la luz del semáforo.

De esta forma, el usuario irá respondiendo a las palabras una por una hasta finalizar el juego.

4.3.3. Resultados

En la siguiente captura, el usuario ha terminado el juego y se muestran los resultados, exponiendo el número de aciertos, fallos y el tiempo en el que se ha finalizado el temporizador al terminar. Además se puede volver a empezar si el usuario lo desea.

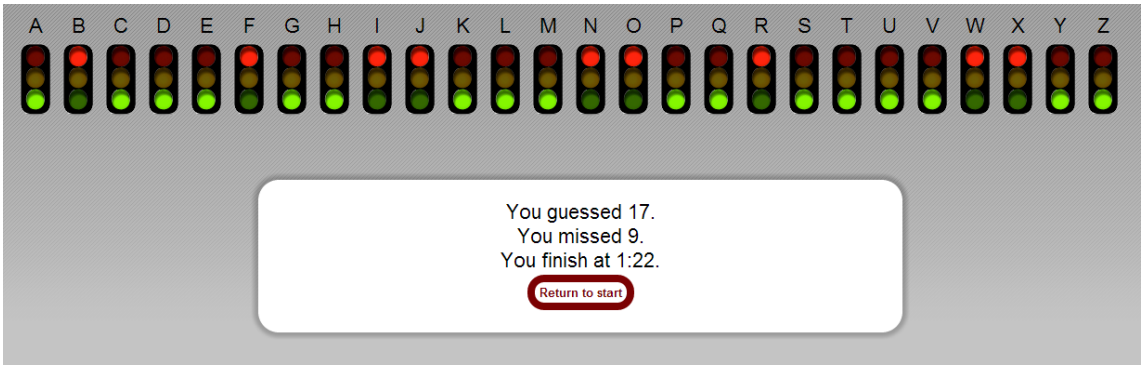


Figura 5.7: Pantalla de resultados de *Traffic Lights*

4.4. Añadir nuevas palabras a la base de datos de la aplicación

Esta aplicación dispone de una base de datos inicial de cinco palabras por letra, y tiene un total de 130 palabras junto con sus definiciones. Esto no quiere decir que sólo se deban utilizar estas palabras, ya que nuestra base de datos una propuesta inicial y es ampliable. Los editores del juego podrán en el futuro, así añadir nuevas palabras a la base de datos. De esta forma, la aplicación puede realizar un número mayor de combinaciones de la lista aleatoria a la hora de iniciar el juego, haciendo posible un juego en el que no se repitan las mismas palabras una y otra vez, a la vez que sea más divertido para el usuario.

A continuación se explica cómo añadir nuevas palabras a la base de datos de la aplicación. Para ello, el administrador de la aplicación deberá acceder a la base de datos de su servidor web, y teniendo el gestor de contenidos **Wordpress**, se hará desde el panel de administración de Wordpress junto con el plugin ya previamente añadido ***Adminer*.**

En primer lugar, se debe acceder al escritorio de Wordpress del sitio web donde se aloja la aplicación, en nuestro caso el ILLLab (Figura 5.1):

Figura 5.8: Imagen de la pantalla de entrada a WordPress

Introduciremos el usuario y la contraseña del administrador y entraremos al escritorio de administración del gestor de contenidos. Una vez ahí, veremos una barra lateral a la izquierda (Figura 5.2) donde se encuentran los menús de las distintas opciones que tiene Wordpress. Nos centraremos en "Tools", que es donde se puede encontrar el plugin ***Adminer***.

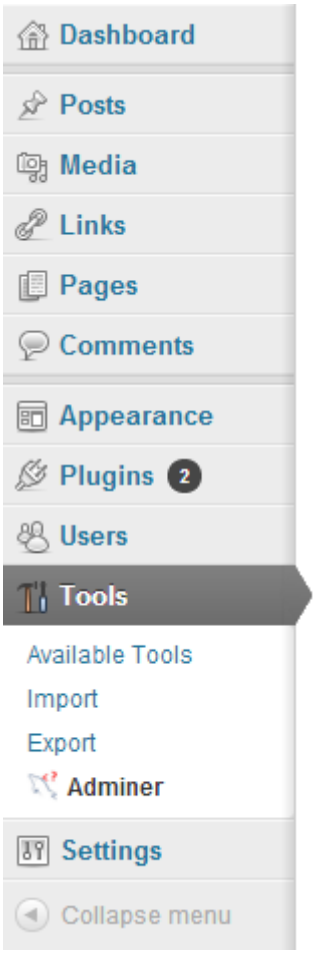


Figura 5.9: Menú de opciones del panel de control de WordPress

Después, aparecerá la pantalla del plugin ***Adminer*** con algunos datos sobre la base de datos del ILLLab y también dos botones con los que se podrá entrar en la base de datos para poder crear/borrar tablas, importarlas o exportarlas, modificarlas, etc. Cualquiera de los dos botones permitirá al administrador acceder a la base de datos, la diferencia de uno a otro es que el primero lo abre dentro del escritorio de Wordpress y el segundo lo abre en una pestaña diferente del navegador.

Figura 5.10: Pantalla de inicio del plugin *Adminer*

Al pulsar el botón, entraremos a la base de datos, donde se puede observar todas las tablas que contiene ILLLab, en la imagen a continuación mostraremos un ejemplo de ello.

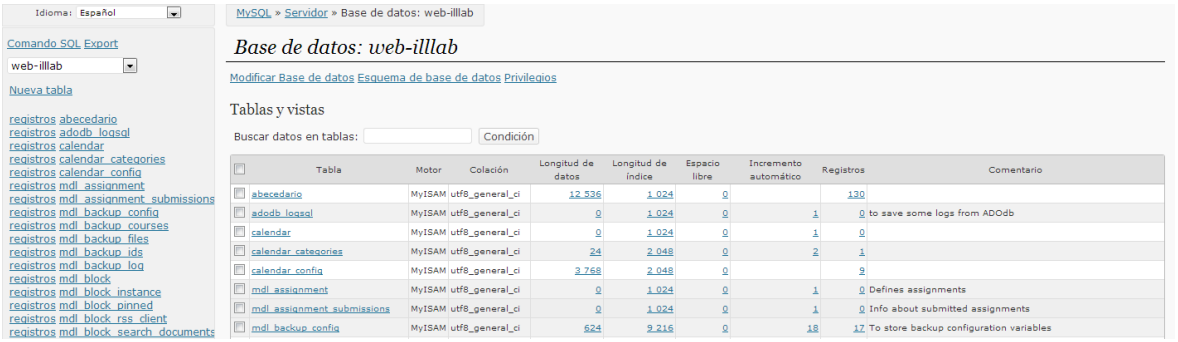


Figura 5.11: Pantalla de la base de datos de WordPress

La tabla que se utiliza para la aplicación se llama ***abecedario***, que es donde se almacena todas las palabras y definiciones para que la aplicación funcione correctamente. Se procederá a su búsqueda y en la siguiente captura se puede observar su estructura y parte de su contenido:

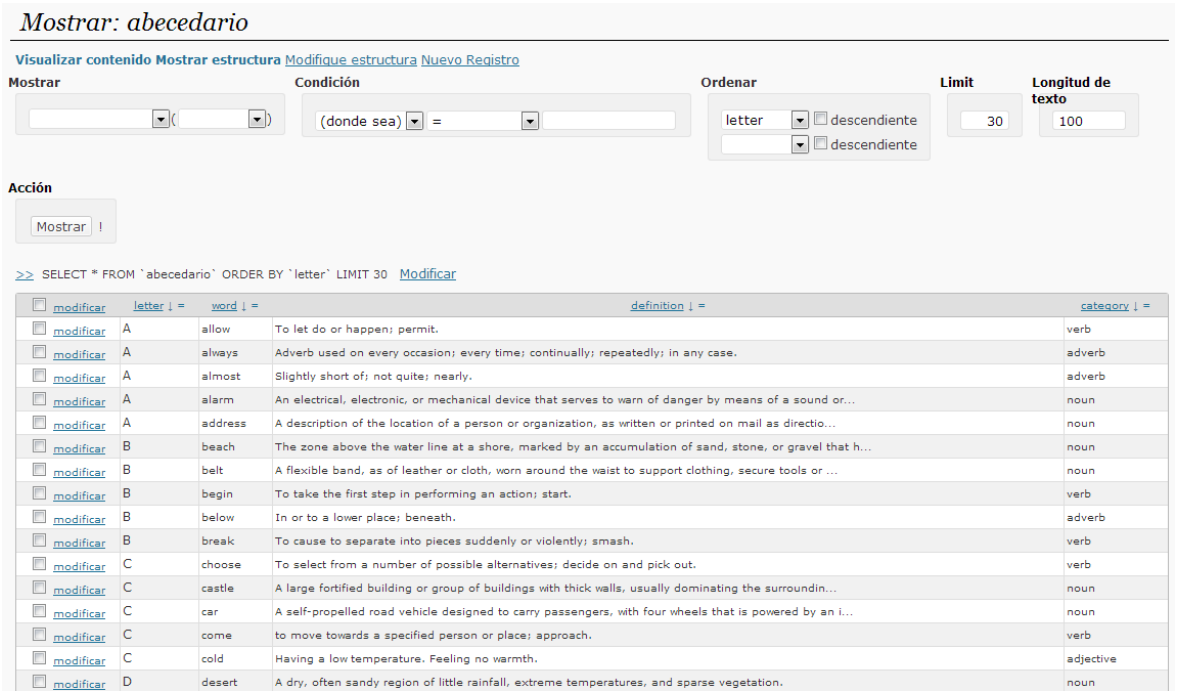


Figura 5.12: Visualización del contenido de la tabla "abecedario"

La tabla se puede modificar y es posible añadir nuevos campos, de manera que se puedan realizar futuras mejoras de la aplicación y usar los nuevos campos en estas nuevas actualizaciones. En la captura se puede observar su estructura:

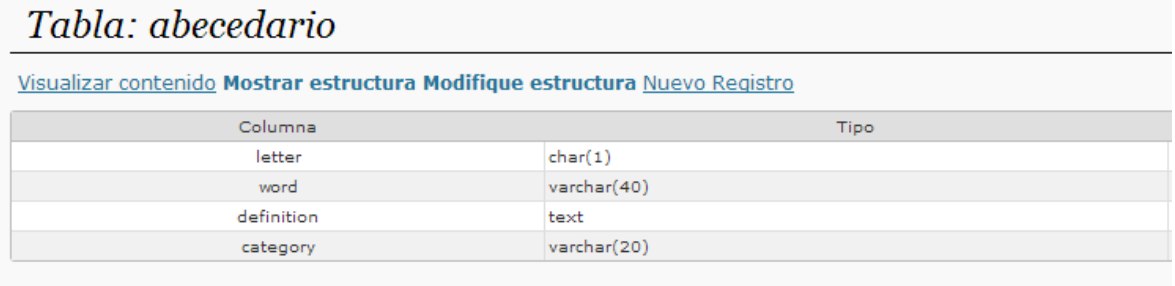


Figura 5.13: Estructura de la tabla "abecedario"

El administrador debe tener en cuenta la estructura de la tabla para poder rellenarla, la estructura es la siguiente:

- ***letter:*** letra por la que empieza la palabra que se quiere añadir. (1 Carácter)
- ***word:*** palabra que se desea añadir. (40 caracteres)
- ***definition:*** definición de la palabra a añadir. (texto indefinido)
- ***category:*** categoría a la que pertenece la palabra. (20 caracteres)

Para añadir una nueva palabra, en la parte superior derecha la de última imagen se puede observar el enlace de *"Nuevo registro"*, si pulsamos en este enlace, irá a la pantalla para introducir un nuevo registro de la nueva palabra que deseamos introducir:

Figura 5.14: Inserción una nueva palabra en la tabla "abecedario"

En esta tabla rellenaremos todos los campos de la estructura de la tabla (letra, palabra, definición y categoría) y la podremos guardar o guardar e insertar otro nuevo registro, es decir, una nueva palabra.

Nota: En la columna del medio donde aparecen unos campos de texto desplegables, no se deberá elegir ninguna opción de las que ponen porque sirven para codificar la información introducida en ese campo, y no es necesario codificarla ya que carece de importancia comparado con contraseñas y otros campos de interés para el administrador. La codificación consiste en sustituir unidades textuales con importancia semántica para ocultar la información.

Ahora se explicará que como debe rellenarse cada uno de los campos de la estructura de la tabla para que no provoque fallos en la aplicación:

- En el campo *letter* se pondrá una letra **mayúscula** de entre la A y la Z, no debe exceder de un carácter.
- En el campo *word* se escribirá una palabra que no exceda el máximo de caracteres, en este caso 40.
- En el campo *definition* se podrá escribir un número de caracteres indefinido.
- En el campo *category*, se asignará una categoría gramatical de las siguientes:
    - noun
    - verb
    - adverb
    - adjetive

Si todo ha sido introducido correctamente, el nuevo registro pasará a formar parte de la tabla y la aplicación lo utilizará satisfactoriamente.

De esta forma se finaliza este breve tutorial para añadir una palabra a la base de datos de la aplicación.

4.5. Modificar el tiempo del temporizador

El temporizador con cuenta de atrás actúa como una parte importante del juego, que dificulta al usuario la resolución del juego completo, haciendo que éste sea un verdadero reto. Por esto, se define este tutorial para que el administrador de ILLLab pueda modificar este tiempo ya que, por ejemplo, el administrador puede necesitar cambiarlo debido a que haya un alto porcentaje de usuarios que no puedan completar a tiempo una partida o por otras causas que el administrador estime oportuno.

Por defecto, se ha establecido un tiempo de 300 segundos, es decir, de cinco minutos, para que el usuario averigüe el mayor número de palabras posibles. Para que el administrador pueda cambiarlo debe acceder a un archivo llamado "functions.js" de modo que necesita conectarse vía FTP (File Transfer Protocol) al servidor web donde se encuentra alojada la aplicación.

Filezilla es el cliente FTP que se utiliza en este tutorial para conectarnos al servidor web del ILLLab, aunque cualquier otro cliente FTP es igual de válido que éste.

En primer lugar, explicaremos la interfaz de Filezilla para familiarizar al administrador con el entorno de la aplicación. Empezaremos con la barra del menú superior, en concreto sólo con el botón que nos interesa que marcaremos en rojo para una mejor visualización:

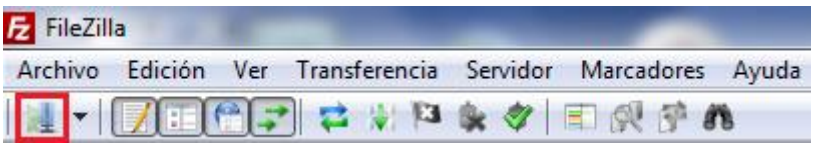


Figura 5.15: Barra de menús de *Filezilla*

Este botón sirve para abrir la ventana del gestor de sitios. En esta ventana es donde rellenaremos los parámetros necesarios para una correcta conexión con el servidor web deseado. Para crear un nuevo sitio web pulsaremos **Nuevo sitio**. Una vez creado, nos centraremos en la pestaña **General** de la ventana, en la que nos encontramos con los siguientes campos que tendremos que rellenar:

- **Servidor:** Es el servidor web donde se encuentra alojado ILLLab y nuestra aplicación.
- **Puerto:** Es el puerto que utiliza el servidor web para su acceso.
- **Protocolo:** Es el protocolo que se utiliza para transferir los archivos. En nuestro caso utilizaremos **SFTP** (Secure File Transfer Protocol).

- **Modo de acceso:** Establece el modo de acceder al servidor web. Puede ser anónimo, normal, preguntar contraseña, interactivo y cuenta. En nuestro caso utilizaremos el modo *normal*, que nos habilitará las opciones de usuario y contraseña.
- **Usuario:** Es el usuario que desea acceder a la base de datos.
- **Contraseña:** Es la contraseña que se utiliza para verificar la autenticidad del usuario y el acceso al servidor web.

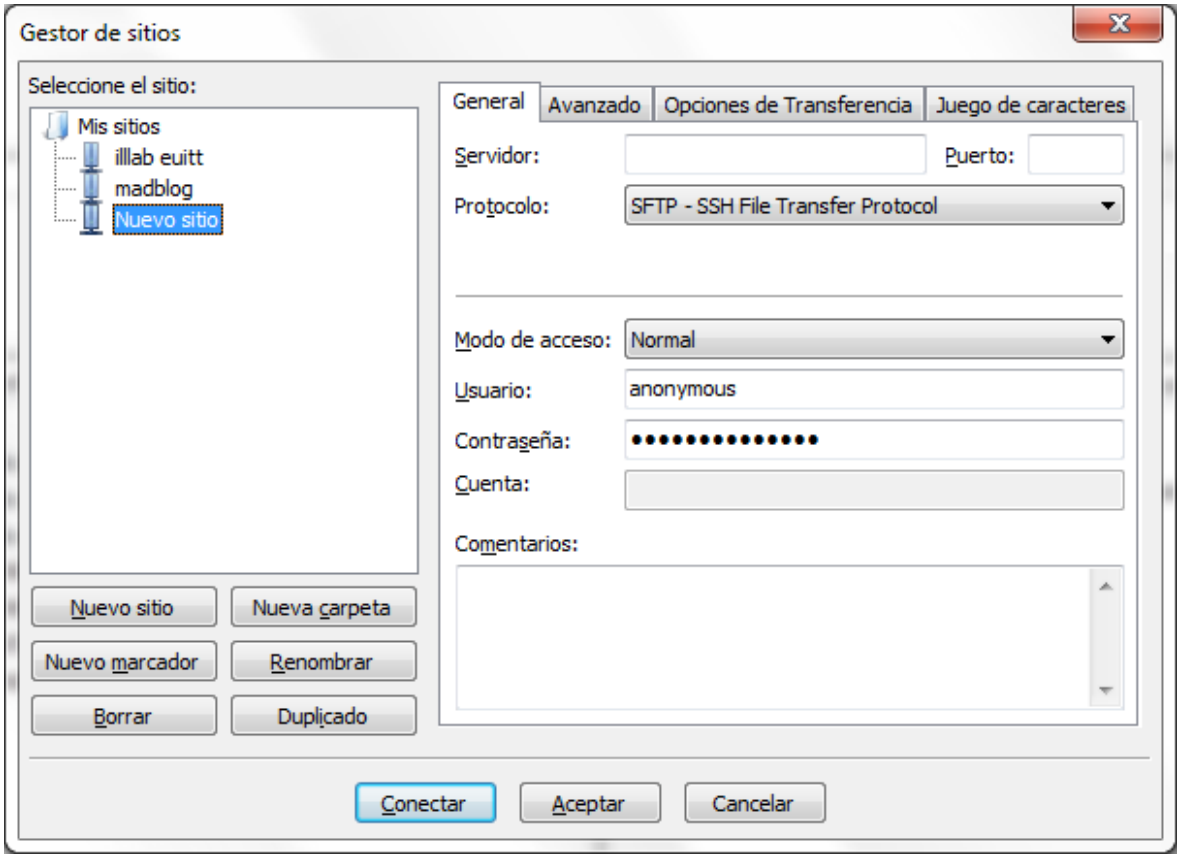


Figura 5.16: Gestor de sitios de *Filezilla*

Nota: No se facilitan los datos de acceso para la seguridad del servidor web, preguntar al administrador del servidor web para adquirir su aprobación y la información necesaria para acceder al servidor web.

Una vez rellenos todos los campos necesarios para el acceso al servidor web, nos conectaremos con el servidor para acceder al fichero que deseamos. Veremos que debajo de la barra de menú hay una ventana que aparecerán los comandos que utiliza el cliente FTP para conectarse al servidor web. Si todo ha salido bien, estaremos conectados al servidor. Debajo de esta ventana de comandos aparecerá una ventana divida verticalmente en dos partes. Esta ventana es un gestor de archivos, a la izquierda nos encontraremos con los archivos locales y a la derecha con los archivos remotos, es decir, con los archivos del servidor web al que estamos conectados.

Desde la ventana de la derecha accederemos al fichero “functions.js” que queremos modificar. Para ello tenemos que navegar por el gestor de archivos hasta encontrarlo. La ruta para acceder al fichero a modificar es la siguiente “*/moodle/course/trafficlights/js”*. Esta ruta indica que dentro de la carpeta *moodle*, encontraremos la carpeta *course*, y dentro de ésta estará *trafficlights*, nuestra aplicación, y por último dentro estará *js*, que contiene los ficheros JavaScript donde encontraremos “functions.js”.

Para poder modificarlo, tendremos que descargarlo del servidor web a nuestro disco duro local, para ello elegimos en la parte de la izquierda del gestor de archivos la ruta donde queremos guardar el fichero “functions.js”, por ejemplo el escritorio, y lo guardamos. Para modificar este archivo se puede utilizar un editor de texto cualquiera, como por ejemplo el ***Bloc de notas*** de Windows, en nuestro caso utilizamos **Crimson Editor**.

A continuación abrimos el fichero y buscamos las siguientes sentencias, situadas por la línea 135 del fichero:

```
135 shortly = new Date();
136 shortly.setSeconds(shortly.getSeconds() + 300);
137 $('#defaultCountdown').countdown('option', {until: shortly, format: 'MS'});
```

Este fragmento de código establece en el temporizador el número de segundos con el que deseamos que empiece y el formato en que queremos que muestre en la aplicación, en este caso MS: minutos y segundos. Sólo se necesita cambiar el número que aparece marcado en rojo por los segundos deseados por el administrador y guardar el fichero.

Volviendo a Filezilla, iremos a la ventana de la izquierda del gestor de archivos y transferiremos nuestro fichero modificado “functions.js” al servidor remoto, de modo que sobrescribiremos el anterior fichero “functions.js”.

Finalmente, probaremos la aplicación para comprobar que el tiempo del temporizador ha sido modificado correctamente.

5. CONCLUSIONES

En este proyecto se ha realizado un juego de autoaprendizaje dirigido a estudiantes de nivel de inglés A2. Las bases de la aplicación han sido los lenguajes de programación web en el cliente, que son HTML, JavaScript, junto con las librerías de efectos y manejo de documentos HTML proporcionadas por jQuery, y la comunicación asíncrona que ofrece AJAX; además de los lenguajes de programación web en el servidor, PHP y MySQL.

Los resultados obtenidos son satisfactorios desde el punto de vista del usuario, puesto que se realizaron pruebas de la aplicación con varios usuarios de diferente nivel de inglés, y sus opiniones fueron de una aplicación muy didáctica y entretenida.

A continuación, expondré algunas conclusiones personales:

Este proyecto ha sido realizado desde los conocimientos obtenidos en la ingeniería técnica de telecomunicaciones en la especialidad de telemática y, que por tanto, me han servido para desarrollar la aplicación que el Departamento de Lingüística me propuso.

En un principio, este proyecto se diseñó pensando aportar ayuda al autoaprendizaje que ofrece la plataforma de ILLLab, lo que me supuso para mí una motivación para desarrollarla, de manera que todo aquel que quiera aprender a través de esta plataforma pueda encontrar un recurso más que brindar al usuario.

El desarrollo del trabajo ha sido un gran reto porque me encontré muchas dificultades al principio, pero con mis propios recursos fui encontrando soluciones a los problemas que tenía y superándolos, consiguiendo resultados satisfactorios. He de destacar que el objetivo no alcanzado ha sido conseguir un reconocedor de voz que permita una experiencia de usuario fluida, puesto que al no reconocer la palabra que el jugador está expresando, hace perder tiempo a la vez que frustra tener que volver intentarlo.

Gracias a este proyecto he ampliado mis conocimientos sobre la tecnología web del momento, así como del reconocimiento y la síntesis de voz, que va ganando un papel cada vez más importante en la tecnología, haciendo sustituir al ser humano por la máquina.

He de decir que la realización de este proyecto ha sido gratificante porque espero que los alumnos puedan utilizar la aplicación para aprender vocabulario de forma amena, además de practicar su pronunciación.

5.1. Mejoras para la aplicación

Este proyecto presenta diversas mejoras, tanto del punto de vista del usuario como del desarrollador de la aplicación.

Desde el punto de vista del usuario se puede mejorar el diseño de la aplicación, como su apariencia, su interfaz o añadir efectos y animaciones que hagan que la aplicación dé una experiencia al usuario, sencilla de manejar a la par que divertida.

Asimismo, se puede añadir más opciones al juego, como la posibilidad de jugar dos o más personas en modo multijugador, haciendo que cada vez que un usuario pase de una palabra o conteste una palabra incorrectamente, se pasa el turno al otro jugador.

Igualmente se pensó en añadir un ranking de puntuación, o de número de palabras acertadas en un tiempo determinado, para que los jugadores lo pudieran consultar y así suponer un desafío intentar superarse uno mismo o a otros jugadores. En un principio, esta opción no la llevamos a cabo debido a que suponía añadir registros a la base de datos y se temía por una brecha de seguridad en la base de datos de ILLLab, que además, forma parte de la base de datos de la UPM y su acceso está restringido externamente. Al no tener conocimientos en el campo de seguridad en bases de datos, desistimos de esta opción.

También se ha pensado la posibilidad de añadir la opción de síntesis de voz, como por ejemplo, que la aplicación lea la definición para que el usuario practique su comprensión oral.

Desde el punto de vista del desarrollador de la aplicación, las mejoras anteriores podrían suponer un reto, pero además se podrían realizar más avances en cuánto a la aplicación.

Por ejemplo, la base de datos comienza con un pequeño número de palabras insertadas, de forma que el administrador de la aplicación puede añadir más palabras con sus definiciones. No obstante, la posibilidad de modificar la estructura de la tabla para mejorar la experiencia del usuario puede hacer una aplicación más interesante. Se pueden añadir nuevos campos a los registros de la base de datos, por ejemplo, poner un nivel de dificultad a las palabras u ordenarlas por familias de palabras, como alimentos, hogar, etc. Y antes de que los usuarios comiencen una partida, éstos podrán escoger el nivel de dificultad suponiendo un mayor reto a mayor nivel, y/o elegir una familia concreta de palabras, para aumentar o repasar su vocabulario.

Finalmente, se puede hacer que el usuario escoja una, varias o todas las opciones descritas anteriormente, según la menor o mayor dificultad que le suponga al usuario practicar con esta aplicación. Por ejemplo, que el jugador escoja sólo síntesis de voz porque se le da mejor la comprensión que la expresión oral o porque precisamente quiere mejorar en ese aspecto.

Por otra parte, el reconocedor de voz no presenta un porcentaje alto de comprensión, en el que confunde determinadas palabras con otras que suenan parecidas. Al ser una herramienta de libre distribución, es posible que haya mejores reconocedores de voz de pago más adaptables a la aplicación. Sin embargo, al no disponer de financiación, se ha encontrado este complemento que hace una buena función para lo que necesitamos. Como futura mejora, posiblemente más adelante aparezcan versiones mejores de este complemento u otros reconocedores de voz gratuitos con un alto índice de comprensión que se pueda implementar en la aplicación.

BIBLIOGRAFIA

HEADWAY: Elementary Student's book

Liz & John Soars,

Oxford University Press

ISBN 10: 0194339920 / 0-19-433992-0

Editorial: Oxford University Press

Fecha de publicación: 1996

Consultas para el desarrollo de la aplicación:

- HTML5: http://www.w3.org/html/logo/
- PHP: http://php.net/manual/es/index.php
- MySQL: http://www.mysql.com/
- jQuey y AJAX: http://jquery.com/
- JSAPI: http://docs.oracle.com/cd/E17802_01/products/products/java-media/speech/forDevelopers/jsapi-doc/
- Talking Java SDK: http://www.cloudgarden.com/JSAPI/
- Ejemplo de desarrollo JSAPI + Talking Java SDK: http://cmop17.wordpress.com/2010/06/09/javareconocimiento-y-sintetizacion-de-voz-con-cloud-garden-talkingjava-sdk-with-java-speech-api-implementation/
- WebSpeech API: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html
  Ejemplo de utilización: http://slides.html5rocks.com/#speech-input
- JSON: http://es.wikipedia.org/wiki/JSON
- DOM: http://www.w3.org/DOM/
  http://es.wikipedia.org/wiki/Document_Object_Model

Consultas de ejemplos y dudas:

- http://www.w3schools.com/
- http://www.desarrolloweb.com/
- http://www.forosdelweb.com/
- http://stackoverflow.com
- http://librosweb.es/

Plugin del temporizador jQuery:

- http://keith-wood.name/countdown.html

Servidor local XAMPP:

- http://www.apachefriends.org/es/xampp.html

Foro oficial WordPress: http://es.forums.wordpress.org/

Adminer, plugin para WordPress: http://www.adminer.org/

Filezilla FTP Cliente: https://filezilla-project.org/

Diccionarios utilizados para la realización de las definiciones:

- http://www.wordreference.com/
- http://www.thefreedictionary.com/
- http://dictionary.cambridge.org/

Reconocimiento de voz:

- http://es.wikipedia.org/wiki/Reconocimiento_del_habla
- http://dihana.cps.unizar.es/investigacion/voz/rahframe.html
- http://www.nuance.es/ucmprod/groups/dragon/@web-es/documents/collateral/nq_020444.pdf
- http://www.verbio.com/webverbio3/index.php
- http://www.tellmemore.com/
- http://cmusphinx.sourceforge.net/
- http://www.microsoft.com/en-us/download/details.aspx?id=10121

APÉNDICE A: Palabras y definiciones integradas en la base de datos

En este apéndice se recopilan todas las palabras junto con sus definiciones que inicialmente se integraron en la base de datos.

DEFINICIONES:

A

Address

A description of the location of a person or organization, as written or printed on mail as directions for delivery.

Alarm

An electrical, electronic, or mechanical device that serves to warn of danger by means of a sound or signal.

Allow

To let do or happen; permit.

Always

Adverb used on every occasion; every time; continually; repeatedly; in any case.

Almost

Slightly short of; not quite; nearly…

B

Beach

The zone above the water line at a shore, marked by an accumulation of sand, stone, or gravel that has been deposited by the tide or waves.

Belt

A flexible band, as of leather or cloth, worn around the waist to support clothing, secure tools or weapons, or serve as decoration.

Begin

To take the first step in performing an action; start.

Break

To cause to separate into pieces suddenly or violently; smash.

Below

In or to a lower place; beneath.

C

Car

A self-propelled road vehicle designed to carry passengers, with four wheels that is powered by an internal-combustion engine.

Castle

A large fortified building or group of buildings with thick walls, usually dominating the surrounding country.

Choose

To select from a number of possible alternatives; decide on and pick out.

Come

To advance toward the speaker or toward a specified place; approach.

Cold

Having a low temperature. Feeling no warmth.

D

Desert

A dry, often sandy region of little rainfall, extreme temperatures, and sparse vegetation.

Desk

A piece of furniture typically having a flat or sloping top for writing and often drawers or compartments.

Day

The period of light between dawn and nightfall; the interval from sunrise to sunset.

Dance

To move rhythmically usually to music, using prescribed or improvised steps and gestures.

Dry

Free from liquid or moisture.

E

Earth

The third planet of the sun.

Engine

A machine that converts energy into mechanical force or motion.

Eat

To take into the body by the mouth for digestion or absorption.

Enjoy

To receive pleasure or satisfaction from. To have a pleasurable or satisfactory time.

Enough

To a satisfactory amount or degree; sufficiently.

F

Fire

A rapid, persistent chemical change that releases heat and light and is accompanied by flame, especially the exothermic oxidation of a combustible substance.

Friend

A person whom one knows, likes, and trusts.

Fight

To attempt to harm or gain power over an adversary by blows or with weapons.

Find

To discover by searching or making an effort something lost, or meet with.

Fast

Acting, moving, or capable of acting or moving quickly; swift.

G

Gift

Something that is bestowed voluntarily and without compensation. A present.

Glove

A fitted covering for the hand with a separate sheath for each finger and the thumb. A gauntlet.

Glue

A strong liquid adhesive obtained by boiling collagenous animal parts into hard gelatin and then adding water.

Give

To make a present of or to place in the hands of; pass.

Gentle

Considerate or kindly in disposition; amiable and tender.

H

Heart

The chambered muscular organ in vertebrates that pumps blood received from the veins into the arteries, thereby maintaining the flow of blood through the entire circulatory system.

Horse

A large hoofed mammal having a short-haired coat, a long mane, and a long tail, domesticated since ancient times and used for riding and for drawing or carrying loads.

Have

To be in possession of; To possess as a characteristic, quality, or function.

Happy

Enjoying, showing, or marked by pleasure, satisfaction, or joy.

Huge

Of exceedingly great size, extent, or quantity.

I

Interview

A formal meeting in person, especially one arranged for the assessment of the qualifications of an applicant.

Iron

A silvery-white, lustrous, malleable, ductile, magnetic or magnetizable, metallic element used alloyed in a wide range of important structural materials.

Imagine

To form a mental picture or image of; To think; conjecture.

Invite

To ask for the presence or participation of; To request formally.

Interesting

Arousing or holding the attention; absorbing.

J

Jeans

Informal trousers for casual wear, made of denim or corduroy.

Joke

Something said or done to evoke laughter or amusement, especially an amusing story with a punch line.

Judge

A public official who hears and decides cases brought before a court of law.

Join

To put or bring together so as to make continuous or form a unit; To become a part or member of.

Jump

To spring off the ground or other base by a muscular effort of the legs and feet.

K

Key

A metal instrument that is turned to open or close a lock.

Kid

A child. A young person.

Knee

The joint between the thigh and the lower leg, formed by the articulation of the femur and the tibia and covered anteriorly by the patella.

Kick

To strike out with the foot or feet.

Kiss

To touch or caress with the lips as an expression of affection, greeting, respect, or amorousness.

L

Law

A rule or body of rules made by legislature.

Letter

A written or printed communication directed to a person or organization.

Learn

To gain knowledge, comprehension, or mastery of through experience or study.

Late

Coming, occurring, or remaining after the correct, usual, or expected time; delayed.

Long

Extending or traveling a relatively great distance. Having relatively great height; tall. Of relatively great duration.

M

Man

An adult male human.

Menu

A list of the dishes to be served or available for a meal.

Monkey

Any primate except man.

Make

To cause to exist or happen; bring about; create.

Meet

To come together.

N

Neck

The part of the body joining the head to the shoulders or trunk.

Night

The period between sunset and sunrise, especially the hours of darkness.

Need

To be in want of or required of.

Near

To, at, or within a short distance or interval in space or time.

New

Having been made or come into being only a short time ago; recent.

O

Omelet

A dish consisting of beaten eggs cooked until set and folded over, often around a filling.

Orange

Any of various reddish yellow, acid or sweet, edible citrus fruits.

Offer

To present for acceptance or rejection. To put forward for consideration; propose.

Order

To issue a command or instruction to.

Old

Having lived or existed for a relatively long time; far advanced in years or life.

P

President

The chief officer of a branch of government, corporation, board of trustees, university, or similar body.

Potato

A type of plant with tuber which has a brown or red skin and are used as a vegetable.

Pay

To give money to in return for goods or services rendered.

Pass

To happen; take place. To go by without stopping; leave behind. To go across; go through.

Perhaps

Maybe; possibly.

Q

Queen

The wife or widow of a king.

Question

An expression of inquiry that invites or calls for a reply. An interrogative sentence, phrase, or gesture.

Quiz

A short oral or written test.

Quit

To depart from; leave.

Quote

To cite or refer to for illustration or proof.

R

Raincoat

A waterproof or water-resistant coat.

Rival

One who attempts to equal or surpass another, or who pursues the same object as another; a competitor.

Read

To examine and grasp the meaning of printed or written characters, as of words or music.

Run

To travel over on foot at a pace faster than a walk.

Right

In a straight line; directly; Exactly; just; In the proper or desired manner.

S

Student

One who is enrolled or attends classes at a school, college, or university.

Shoe

A durable covering for the human foot, made of leather or similar material with a rigid sole and heel, usually extending no higher than the ankle.

See

To perceive with the eye.

Sometimes

Now and then; from time to time; occasionally.

Soon

In the near future; shortly; early.

T

Tea

Any of various beverages, made as by steeping the leaves of certain plants or by extracting an infusion.

Teacher

A person whose occupation is teaching others.

Tell

To communicate by speech; express with words.

Together

In or into contact or union with each other, a single group, mass or place.

Tall

Having greater than ordinary height.

U

Umbrella

A portable device used for protection against rain, snow, etc, and consisting of a light canopy supported on a collapsible metal frame mounted on a central rod.

Uncle

The brother of one's mother or father, or the husband of one's aunt.

Universe

All matter and energy, including the earth, the galaxies, and the contents of intergalactic space, regarded as a whole.

Understand

To perceive and comprehend the nature and significance of; to grasp.

Unknown

Not identified or ascertained; Not known; unfamiliar.

V

Valley

An elongated lowland between ranges of mountains, hills, or other uplands, often having a river or stream running along the bottom.

Video

The visual portion of a televised broadcast.

View

To look at; watch.

Visit

To go to see or spend time at a place with a certain intent. To stay with as a guest.

Vote

To express one's preference for a candidate or for a proposed resolution of an issue.

W

Water

A clear, colorless, odorless, and tasteless liquid, essential for most plant and animal life and the most widely used of all solvents.

Wine

A beverage made of the fermented juice of various kinds of grapes, containing alcohol.

Win

To achieve victory or finish first in a competition.

Well

In a good, skillful, satisfactory or pleasing manner.

Wrong

Not in conformity with fact or truth; incorrect or erroneous.

X

Axe(contiene X)

A hand tool with one side of its head forged and sharpened to a cutting edge, used for felling trees, splitting timber, etc.

Exchange(Contiene X)

To give in return for something received; trade.

Saxophone(Contiene X)

A keyed wind instrument of mellow tone colour, used mainly in jazz and dance music.

Xenophobe

A person who hates or fears foreigners or strangers.

Xylophone

A percussion instrument consisting of a mounted row of wooden bars graduated in length to sound a chromatic scale, played with two small mallets.

Y

Year

A period of twelve months from any specified date, such as one based on the four seasons.

Yellow

A color like that of egg yolk, ripe lemons, etc. the primary color between green and orange in the visible spectrum.

Yesterday

The day before the present day.

Yell

To cry out loudly, as in pain, fright, surprise, or enthusiasm.

Young

Being in an early period of life, development, or growth.

Z

Zero

A cardinal number indicating the absence of any or all units under consideration.

Zipper

A fastening device consisting of parallel rows of metal, plastic, or nylon teeth on adjacent edges of an opening that are interlocked by a sliding tab.

Zone

An area or a region distinguished from adjacent parts by a distinctive feature or characteristic.

Zoo

A place where live animals are kept, studied, bred, and exhibited to the public.

Zoom

To examine more closely; focus on.

APÉNDICE B: Glosario de términos utilizados

– **API.** En ingles Application Programming Interface. Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

–**AJAX.** En ingles Asynchronous JavaScript And XML. Es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones de ejecutan en el cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

– **CMS.** En inglés Content Management System o sistema de gestión de contenidos. Es una interfaz que permite controlar la administración y creación de contenidos de una página web. Permite manejar independientemente contenido y diseño.

– **CSS.** En ingles Cascading Style Sheets, u hojas de estilo en cascada. Se usan para definir el formato de presentación de un documento con marcado HTML o XML.

– **CSV.** En ingles Comma-Sepparated Values, o archivos separados por comas. Es un tipo de documento con un formato abierto y sencillo que sirve para representar datos en tablas. Las columnas se separan por comas o punto y coma.

– **FTP.** En ingles File Transfer Protocol, o protocolo de transferencia de archivos. Es un protocolo utilizado para descargar archivos o subirlos a un servidor, que se apoya en los servicios proporcionados por TCP.

– **HTML.** En ingles HyperText Markup Language, o lenguaje de marcado de hipertexto. Es un lenguaje de marcado utilizado en la elaboración de páginas web, para describir la estructura de la información que contienen en forma de texto.

– **MySQL.** Es un motor para la gestión de bases de datos, multihilo y multiusuario. Se ofrece como software libre, aunque también existe la posibilidad de comprar licencias específicas para un uso corporativo específico no contemplado bajo esta licencia. Es ampliamente utilizado en aplicaciones web, sobretodo en conjunción con PHP en la mayoría de los casos.

– **PHP.** Es un lenguaje de programación diseñado originalmente para el desarrollo web, para generar contenidos dinámicos en el lado del servidor. Puede usarse en la mayoría de servidores web, al igual que en la mayoría de sistemas operativos y plataformas.

– **Plugin.** Es un programa o módulo software que puede anexarse a otra aplicación, sin alterar sus funciones básicas, con objeto de aumentar su funcionalidad, proporcionando algún servicio añadido.

– **WordPress.** Es un sistema de gestión de contenido o CMS, enfocado hacia la creación de blogs u otros sitios web actualizados periódicamente. Se caracteriza por estar realizado usando PHP y MySQL. Dada su facilidad de uso, es ampliamente utilizado para el desarrollo web.

APÉNDICE C: Código fuente utilizado en la elaboración de la aplicación

C.1 Código fuente de la página principal "index.html"

```
<!DOCTYPE html>

<html lang="en">



<head>

   <meta charset="UTF-8" />

   <title>Traffic Lights</title>

   <link rel="stylesheet" type="text/css" href="css/style.css" />

   <link rel="stylesheet" type="text/css" href="css/jquery.countdown.css" />

   <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>

   <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>

   <script type="text/javascript" src="js/jquery.countdown.js"></script>

   <script type="text/javascript" src="js/functions.js"></script>

</head>



<body>

   <div id="container">

      <h1 id="title">TRAFFIC LIGHTS</h1>

      <div id="way">

         <ul>

            <li>A<img id="tfA" src="img/small-apagado.png" alt="semaforo apagado" /></li>

            <li>B<img id="tfB" src="img/small-apagado.png" alt="semaforo apagado" /></li>
```

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

```
        <li>Z<img id="tfZ" src="img/small-apagado.png" alt="semaforo apagado" /></li>
    </ul>
</div><!-- way ends -->


<div id="defaultCountdown" ></div>


<div id="form">
    <form id="startMenu">
      <span id="browser"></span>
       <span id="instructions">
          <p style="text-indent: 1em">
          Traffic Lights is based in the alphabet game, which you must
          to guess the word behind the definition for each letter,
          since A until Z. There are two options:
          </p>
          <p style="margin-left: 1.5em">
             - You can <b>PASS</b> a word that you don't know, the
             traffic light will turn <b style="color: yellow">YELLOW
             </b> and you can answer it in the next round.
          </p>
          <p style="margin-left: 1.5em">
             - You can answer the word in word text and <b>SUBMIT</b>
             it and the game will return two possible replies:
```

```
</p>
<p style="margin-left: 3em">
      - When you guess a word, the traffic light will turn
      <b style="color: green">GREEN</b> and will add five
      seconds to your timer.
</p>
<p style="margin-left: 3em">
   - When you fail a word, the traffic light will turn
   <b style="color: red">RED</b> and it will subtract 5
   seconds to your timer.
</p>
<p>
In all cases, the game move on to the next letter of the
alphabet. The game ends when the timer reach 0, or when you
answer all the words in the alphabet.
</p>
<p>
In addition, you can choose reply the answer writting in the
text box or with recognition voice through the microphone
icon, You can also listen the definition through the speaker
icon.
</p>
<p style="text-align: center;">
```

```
            Press <b>START</b> to begin playing.
            </p>
        </span>
        <input type="button" id="instruct" value="Instructions">
        <input type="button" id="start" value="START" />
    </form>

    <form id="trafficLight">
        <b class="title">Letter: </b><span id="letter"></span>
        <br>
        <b class="title">Definition: </b> <span id="def"></span>
        <br>
        <input name="wordSpeech" id="wordSpeech" type="text" x-webkit-speech
autocomplete="off" />
        <input type="button" id="pass" value="PASS" />
        <input type="button" id="submit" Value="SUBMIT" />
        <br>
    </form>
    <br>
    <div id="end">
        <span id="green"></span><br>
        <span id="red"></span><br>
        <span id="time"></span><br>
```

```
            <input type="button" id="return" Value="Return to start"
onClick="self.location='index.html'" />

          </div>

          <div id="error">

            <span id="msgError"></span>

            <br/>

            <input type="button" id="return" Value="Return to start"
onClick="self.location='index.html'" />

          </div>

       </div><!-- form ends -->

    </div><!-- container ends -->


</body>


</html>
```

C.2 Código fuente de la página de procesamiento “trafficlights.php”

```
<?php


session_start();

include("dbfunctions.php");

include("functions.php");




$letterlast =  '';


$start = sGetParam("start");

if (isset($start)){

   $letter = 'A';


   if(!(connect_db()))

      die ("Error: No se ha podido conectar a la db");

   else{

      $list = obtain_randlist_word();

      close_db();

   }

   $_SESSION['list'] = $list; //nueva lista

   $_SESSION['round'] = 0; //nº de vueltas

   $_SESSION['end'] = 26; //nº letras hasta finalizar
```

```
}else{
   $list = $_SESSION['list'];

   $letter = sGetParam("letter"); // letra a la que se ha contestado o pasado
   $pass = sGetParam("pass");
   $wordSpeech = sGetParam("wordSpeech");

   $end = sGetParam("end");
   if(isset($end)){
      $_SESSION['end'] = 0;
   }

   if(!isset($pass)&&!isset($end)){
      if($wordSpeech==$list[$letter]['word']){
         $list[$letter]['state']=GREEN;
      }else{
         $list[$letter]['state']=RED;
      }
      $_SESSION['end']--;//Contestamos a una letra
   }

   $letterlast = $letter; //última letra a la que se ha contestado
   $letter++; // SE PASA A LA SIGUIENTE LETRA
```

```
if($letter=='AA'){
    $letter = 'A';
    $_SESSION['round']++;
}
if($_SESSION['round']!=0&&$_SESSION['end']!=0){
    while(( $list[$letter]['state']!=YELLOW )&&( $_SESSION['end']!=0 )){
        $letter++;
        if($letter=='AA'){
            $letter = 'A';
            $_SESSION['round']++;//Realmente esto da = si no hay un maximo de vueltas.
        }
    }
}
if($_SESSION['end']==0){
    $countGREEN = 0;
    $countRED = 0;

    for($letterFail='A'; $letterFail!='AA'; $letterFail++){
        if($list[$letterFail]['state']==GREEN)
            $countGREEN++;
        if($list[$letterFail]['state']==RED)
            $countRED++;
```

```
        }


        session_unset();

        session_destroy();

        $parametros_cookies = session_get_cookie_params();

        setcookie(session_name(),0,1,$parametros_cookies["path"]);


        die(json_encode(array("countGREEN" => $countGREEN, "countRED" => $countRED, "state"
=> $list[$letterlast]['state'],"letterlast" =>$letterlast, "end" => "end")));

    }

}


$_SESSION['list'] = $list;

session_write_close();


if($letterlast == '')

die (json_encode(array("letter" => $letter, "definition" => $list[$letter]['definition'],"letterlast"
=>$letterlast,"fin" => $_SESSION['end'])));

else

die (json_encode(array("letter" => $letter, "definition" => $list[$letter]['definition'], "state" =>
$list[$letterlast]['state'],"letterlast" =>$letterlast,"fin" => $_SESSION['end'])));

?>
```

C.3 Código fuente de la página de funciones php de tratamiendo de la base de datos mysql “dbfunctions.php”

Nota: Los valores de las constantes de la base de datos han sido sustituidas por unas ficticias para que no haya problemas indeseados.

```
<?php

/***** CONSTANTS *****/

DEFINE("SERVER","server");

DEFINE("USERNAME","user");

DEFINE("PASSWORD","password");

DEFINE("DATABASE","wp_illlab");


DEFINE("RED",0);

DEFINE("GREEN",1);

DEFINE("YELLOW",2);


function connect_db() {

  $link=   mysql_connect(SERVER,USERNAME,PASSWORD);

  $connection= mysql_select_db(DATABASE,$link);

  return $link && $connection;

} // connect_db


function close_db() {

  mysql_close();

} // close_db
```

```
function obtain_randlist_word(){
   $letter = 'A';
         //cuando $letter llega a Z y se suma uno mas pasa a AA
   while($letter!='AA'){
      $res_count= mysql_query("SELECT COUNT(*)
                     FROM `abecedario`
                     WHERE letter = '".$letter."'");
      if (!$res_count) {
         echo 'No se pudo ejecutar la consulta: ' . mysql_error();
         exit;
      }
      $rowC = mysql_fetch_row($res_count);
      $sql = "SELECT *
            FROM abecedario
            WHERE letter = '".$letter."'
            LIMIT ".mt_rand(0, $rowC[0]-1)." , 1";
      $result = mysql_query($sql);
      if (!$result) {
         echo 'No se pudo ejecutar la consulta: ' . mysql_error();
         exit;
      }
```

```
        while ($row = mysql_fetch_array($result))
        {
            $list[$letter]['word']=$row['word'];
            $list[$letter]['definition']=$row['definition'];
            $list[$letter]['state']=YELLOW;
        }
        $letter++;
    }
    return $list;
}


?>
```

C.5 Código fuente de la página de funciones javascript “functions.js”

```
$(document).ready(function (){

   $("#trafficLight").hide();

   $("#end").hide();

   $("#instructions").hide();

   $("#way").hide();

   $("#defaultCountdown").hide();

   $("#error").hide();



   $("#instruct").click(instructions);

   $('#start').click(start);

   $('#pass').click(pass);

   $('#submit').click(submit);



   var shortly = 0;

   $('#defaultCountdown').countdown({until: shortly, format: 'MS', onExpiry: end});

   $('input#wordSpeech').keypress(function(e){

      if(e.which == 13){

         /*submit(); /*por si quiero habilitar el ENTER para el submit;

                              NOTA: Puede enviarse 2 veces o mas antes de procesarse */

         return false;

      }

   });
```

```
 if (document.createElement("input").webkitSpeech === undefined) {

      document.getElementById("browser").innerHTML += "<p style='text-align: center;'>Sorry, your browser isn't compatible. Speech input has been implemented in Chrome 11. You can START without speech recognition.</p>";

   }else{

      document.getElementById("browser").innerHTML += "<p style='text-align: center;'>You're using Chrome, Click START button and then click the input box's microphone icon and pronounces the word.</p>";

   }

});
```

```
/***** FUNCIONES AUXILIARES *****/

function addSec(){

   var periods = $('#defaultCountdown').countdown('getTimes');

   var secActual = $.countdown.periodsToSeconds(periods);


   var suma = new Date();

   suma.setSeconds(suma.getSeconds() + secActual + 5);

   $('#defaultCountdown').countdown('option', {until:suma});

}


function subSec(){

   var periods = $('#defaultCountdown').countdown('getTimes');

   var secActual = $.countdown.periodsToSeconds(periods);


   var resta = new Date();

   resta.setSeconds(resta.getSeconds() + secActual - 5);

   $('#defaultCountdown').countdown('option', {until:resta});

   if((resta.getSeconds() + secActual - 5)<=0){

      end();

   }

}
```

```
function instructions(){
   $("#instructions").toggle();
}

function divEnd(countGREEN, countRED){
   document.getElementById("green").innerHTML = "You guessed " + countGREEN + ".";
   document.getElementById("red").innerHTML = "You missed " + countRED + ".";
   $("#trafficLight").hide();
   $("#end").fadeIn(1000);
}

function changeTF(state, letter){
   var tfLetter = "#tf" + letter;
   if(state==0){ //RED
      $(tfLetter).attr("src","img/small-rojo.png");
   }
   if(state==1){ //GREEN
      $(tfLetter).attr("src","img/small-verde.png");
   }
   if(state==2){ //YELLOW
      $(tfLetter).attr("src","img/small-ambar.png");
   }
}
```

```
/***** FUNCIONES PRINCIPALES *****/

function start() {

   $('#start').attr("disabled", "disabled");

   $("#way").show();

   $("#defaultCountdown").show();

   $("#title").hide();

   var start = $("#start").val();

   var dataString = 'start=' + start;


   $.ajax({

      url: 'trafficLights.php',

      type: 'POST',

      data: dataString,

      dataType: "json", /* Esto es importante para que coja los datos

         * como json aunque supuestamente jquery debería reconocerlo,

         * en este caso no lo esta haciendo y me daba problemas */

      success: function(data){

         $('#letter').html(data.letter);

         $('#def').html(data.definition);


         shortly = new Date();

         shortly.setSeconds(shortly.getSeconds() + 240);

         $('#defaultCountdown').countdown('option', {until: shortly, format: 'MS'});
```

```
            $("#startMenu").hide();
            $("#trafficLight").show();
            $("#wordSpeech").focus();
            $('#start').removeAttr("disabled");
        },
        error: function(msg) {
            $('#start').removeAttr("disabled");
            alert(msg);
        }
    });
}
```

```
function submit() {
   $('#pass').attr("disabled", "disabled");
   $('#submit').attr("disabled", "disabled");
   var letter = document.getElementById("letter").innerHTML;
   var wordSpeech = $("#wordSpeech").val().toLowerCase();
   var dataString = 'letter=' + letter + '&wordSpeech=' + wordSpeech;

   $.ajax({
      url: 'trafficLights.php',
      type: 'POST',
      data: dataString,
      dataType: "json",
      success: function(data){
         $('#pass').removeAttr("disabled");
         $('#submit').removeAttr("disabled");
          /* TERMINA EL JUEGO CONTESTANDO A TODAS LAS LETRAS */
         if(data.end=="end"){
            /* FINALIZAR CONTADOR */
            divEnd(data.countGREEN, data.countRED);
            var periods = $('#defaultCountdown').countdown('getTimes');
            if(periods[5]==null && periods[6]==0){
               document.getElementById("time").innerHTML = "You finish at 0.";
            }else{
```

```
                document.getElementById("time").innerHTML = "You finish at " + periods[5] + ":" +
(periods[6] < 10 ? '0' : '') + periods[6] + ".";
            }
            $('#defaultCountdown').countdown('destroy');
            $('#defaultCountdown').hide();
            changeTF(data.state, data.letterlast);
        }else{
            document.getElementById("letter").innerHTML = data.letter;
            document.getElementById("def").innerHTML = data.definition;
            $("#wordSpeech").val("");
            $("#wordSpeech").focus();
            if(data.state==1){
                addSec();
            }else{
                subSec();
            }
            changeTF(data.state, data.letterlast);
        }
    },
    error: function(msg) {
        $('#pass').attr("disabled", "");
        $('#submit').attr("disabled", "");
        $("#trafficLight").hide();
                    $("#way").hide();
```

```
                    $('#defaultCountdown').countdown('destroy');

                    $("#defaultCountdown").hide();

        document.getElementById("msgError").innerHTML = "An error has occurred when the
result is sent";

        $('#error').show();

      }

   });

}
```

```
function pass() {
   $('#pass').attr("disabled", "disabled");
   $('#submit').attr("disabled", "disabled");
   var letter = document.getElementById("letter").innerHTML;
   var dataString = 'letter=' + letter + '&pass=pass';

   $.ajax({
      url: 'trafficLights.php',
      type: 'POST',
      data: dataString,
      dataType: "json",
      success: function(data){
         $('#pass').removeAttr("disabled");
         $('#submit').removeAttr("disabled");
         document.getElementById("letter").innerHTML = data.letter;
         document.getElementById("def").innerHTML = data.definition;
         $("#wordSpeech").val("");
         $("#wordSpeech").focus();
         changeTF(data.state, data.letterlast);
      },
      error: function(msg) {
         $('#pass').attr("disabled", "");
         $('#submit').attr("disabled", "");
```

```
        $("#trafficLight").hide();

                $("#way").hide();

                $('#defaultCountdown').countdown('destroy');

                $("#defaultCountdown").hide();

        document.getElementById("msgError").innerHTML = "An error has occurred when you pass
a word";

        $('#error').show();

      }

   });

}
```

```
function end() {
    $('#pass').attr("disabled", "disabled");
    $('#submit').attr("disabled", "disabled");
    var letter = document.getElementById("letter").innerHTML;
    dataString = 'letter=' + letter + '&end=end';



    $.ajax({
        url: 'trafficLights.php',
        type: 'POST',
        data: dataString,
        dataType: "json",
        success: function(data){
            divEnd(data.countGREEN, data.countRED);
            document.getElementById("time").innerHTML = "You finish at 0.";
            $('#defaultCountdown').countdown('destroy');
            $('#defaultCountdown').hide();
            $('#pass').attr("disabled", "");
            $('#submit').attr("disabled", "");
        },
        error: function() {
            $('#pass').attr("disabled", "");
            $('#submit').attr("disabled", "");
```

```
                    $("#trafficLight").hide();

                    $("#way").hide();

                    $('#defaultCountdown').countdown('destroy');

                    $("#defaultCountdown").hide();

        document.getElementById("msgError").innerHTML = "An error has occurred at the end";

        $('#error').show();

      }

   });

}
```

C.6 Código fuente de la hoja de estilos “style.css”

```
body{

        background: #c4c4c4 url('../img/bck-main.png') repeat-x;

   text-align: center;

}



#container{

   width: 1300px;

   margin: 0px auto;

}

/************ TITLES ************/

h1{

   color: #7b0102;

   font-family: sans-serif;

   font-size: 50px;

   font-weight: bold;

}



/************ CLASSES ************/

.title{

   color: #7b0102;

   text-align: left;

}
```

```
/************ WAY ************/

#way{

   height: 140px;

   text-align: center;

   list-style:none;

}



#way ul{

   margin: 0px auto;

   height: 135px;

   list-style:none;

}



#way li{

   color: black;

   font-family: sans-serif;

   font-size: 22px;

   text-align: center;

   margin:2px;

   padding:2px;

   float:left;

}
```

```
#way li img{
   padding:0px;
   display: block;
}

/************ COUNTDOWN ************/

#defaultCountdown{
            color: #7b0102;
   margin: 1em auto; /* Con auto se CENTRA cualquier objeto! */
   background-color: #f3f3f3;
   font-family: sans-serif;
   font-size: 22px;
   width: 240px;
   height: 60px;
   clear:both;
}
```

```
/************ FORMS ************/

form#trafficLight, form#startMenu, div#end{

   color: black;

   font-family: sans-serif;

   font-size: 22px;

   background: white;

   border-radius: 1em;

   -moz-box-shadow: 0 0 5px 5px rgba(0,0,0,0.2);/* Mozilla */

   box-shadow: 0 0 5px 5px rgba(0,0,0,0.2);

   margin: 1em auto;

   padding: 1em;

   width: 50%;

}



form#trafficLight, div#end{

   text-align: center;

}

form#startMenu{

   text-align: left;

}



form#trafficLight input, form#startMenu input{

   display: block;
```

```
    font-size: 18px;
    margin: 0.5em auto;
    width: 30%;
}

input[type="button"]{
    color: #7b0102;
    font-weight: bold;
    padding: 3px 5px;
    background: white;
    border:8px solid #7b0102;
    border-radius: 25px;
}

/************ BUTTON ************/

input[type="button"]:hover, input[type="button"]:focus {
    background: #FF2418;
    color: white;
    text-decoration: none;
}
```

```
input[type="button"]:active{
   -webkit-box-shadow: inset 0 1px 4px rgba(0, 0, 0, 0.6);
   -moz-box-shadow: inset 0 1px 4px rgba(0, 0, 0, 0.6);
   box-shadow: inset 0 0 8px rgba(0, 0, 0, 0.6);
   background: #FF2418;
   color: white;
}

input[type="button"]:disabled{
   background: black;
   color: white;
}
```

C.7 Código fuente de la hoja de estilos para el temporizador "jquery.countdown.css"

```
/* jQuery Countdown styles 1.6.1. */

.hasCountdown {

	border:8px solid #7b0102;

      border-radius: 25px;

	background-color: #eee;

}

.countdown_rtl {

	direction: rtl;

}

.countdown_holding span {

	color: #7b0102;

}

.countdown_row {

	clear: both;

	width: 100%;

	padding: 0px 2px;

	text-align: center;

}

.countdown_show1 .countdown_section {

	width: 98%;

}

.countdown_show2 .countdown_section {
```

```
        width: 48%;
}
.countdown_show3 .countdown_section {
        width: 32.5%;
}
.countdown_show4 .countdown_section {
        width: 24.5%;
}
.countdown_show5 .countdown_section {
        width: 19.5%;
}
.countdown_show6 .countdown_section {
        width: 16.25%;
}
.countdown_show7 .countdown_section {
        width: 14%;
}
.countdown_section {
        display: block;
        float: left;
        font-size: 75%;
        text-align: center;
}
```

```
.countdown_amount {
        font-size: 200%;
}
.countdown_descr {
        display: block;
        width: 100%;
}
```